TESIS DE MAESTRÍA

DISEÑO DE UN MODELO DE REPROGRAMACIÓN DE LA PRODUCCIÓN BAJO CONDICIONES DE INCERTIDUMBRE EN EL PROCESO DE APROVISIONAMIENTO DE MATERIAS PRIMAS.

KAREN YESENIA QUINTERO PARRA

DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

GRUPO DE INVESTIGACIÓN DE PRODUCTIVIDAD Y COMPETITIVIDAD

ENERO 2022

DISEÑO DE UN MODELO DE REPROGRAMACIÓN DE LA PRODUCCIÓN BAJO CONDICIONES DE INCERTIDUMBRE EN EL PROCESO DE APROVISIONAMIENTO DE MATERIAS PRIMAS.

KAREN YESENIA QUINTERO PARRA

TRABAJO DE GRADO PRESENTADO COMO REQUISITO PARCIAL PARA OPTAR EL TÍTULO DE MAGÍSTER EN INGENIERÍA INDUSTRIAL

DIRECTOR:

Ph. D ALCIDES RICARDO SANTANDER MERCADO

UNIVERSIDAD DEL NORTE

PROGRAMA DE INGENIERÍA INDUSTRIAL

MAESTRÍA EN INGENIERÍA INDUSTRIAL

BARRANQUILLA

ENERO DE 2020

AGRADECIMIENTOS

A Dios, por brindarme sabiduría y guía durante todo el desarrollo de esta investigación. Gracias por darme fuerza y esperanza cuando no tenía ninguna en medio de las dificultades que se presentaron. **Isaías 40:29**

A mi familia, a mis padres Honorio y Mary, y a mis hermanos Carlos y Mary por su apoyo, comprensión y animo durante este proceso de mi formación como magíster.

A mis amigos y hermanos de la fe, que oraron conmigo y me apoyaron y me animaron para finalizar este reto para la gloria de Dios.

Tabla de contenido

Tabla de figuras

Lista de tablas

CAPITULO I: ANTECEDENTES DE LA INVESTIGACIÓN

En esta sección se planteará una descripción general del proyecto de investigación, haciendo énfasis en su relevancia e impacto en la productividad de las organizaciones. Además, se presentará los objetivos y la metodología a seguir de la presente investigación y, por último, el cronograma de actividades.

1. INFORMACIÓN GENERAL DEL PROYECTO

Título de la Investigación: Diseño de un modelo de reprogramación de la producción bajo condiciones de incertidumbre en el proceso de aprovisionamiento de materias primas.		
Nombre del Estudiante	**Código**	**Programa**
Karen Yesenia Quintero Parra	200020013	Maestría en Ingeniería Industrial

Proyecto Adscrito al Grupo (s)	**Director del Proyecto**
Productividad y Competitividad	Alcides R. Santander Mercado

Financiación de Desembolsos en Efectivo			
Proyecto o Empresa	**Director**	**Grupo**	**Monto**

Palabras Clave			**Duración**

¿Esta propuesta está asociada a proyectos en elaboración para financiación interna o externa?

Si □ No ■. De ser afirmativa su respuesta, por favor, indique.

Proyecto	**Convocatoria o Empresa**	**Director del Proyecto**

2. RESUMEN DEL PROYECTO

El curso de esta investigación permitió desarrollar un modelo de reprogramación de la producción como respuesta a una interrupción ocasionada por la no disponibilidad de materia prima de sus tareas o trabajos, con el objetivo de obtener una nueva programación lo más cercana posible a la programación inicial obteniendo resultados similares a las métricas de Makespan y Tardanza total. Para obtener este modelo, se desarrolló en MATLAB un modelo inicial de programación para Flow shop para minimización de Makespan en base al algoritmo genético de Chu-Beasley; luego, en base al modelo anterior, se desarrolló un modelo de programación para Flexible Flow Shop para la solución de un problema bi-objetivo (minimización del Makespan y Tardanza total), y en base a éste, se logró desarrollar el modelo de reprogramación de la producción que minimiza la sumatoria de la diferencia de los tiempos de inicio de los trabajos entre la programación inicial y la nueva secuencia, y a su vez, minimiza la tardanza total.

Con la literatura revisada en esta investigación se logró 1) Identificar los elementos con incertidumbre que pueden afectar un plan de producción donde se evidencia que existen pocas investigaciones referentes a lead time con presencia de sólo un 3% , 2) Identificar el tipo de ambiente de programación de cada investigación, donde se evidenció que para el caso de reprogramación en Flexible Flow Shop (FFS) existen pocas investigaciones con presencia de sólo un 4%, 3)Identificar el método de solución utilizado para cada ambiente de programación.

Con respecto a las investigaciones de reprogramación de la producción para Flexible Flow Shop, se puede destacar que ninguna de estas investigaciones considera el lead time como el tipo de incertidumbre que afecta su plan de producción; y en lo que se refiere a su función objetivo, aunque incluyen minimización de Makespan o tardanza, ninguno considera una función bi-objetivo que permita la minimización de estas dos métricas. Por lo anterior, el modelo de reprogramación propuesto en esta investigación se entiende como una contribución a la literatura para ambientes de programación de Flexible Flow Shop bajo condiciones de incertidumbre en el proceso de aprovisionamiento de materias primas ya que permite mejorar

la confiabilidad sobre las métricas de Makespan y de tardanza del programa de producción asociado.

Finalmente, para evaluar el desempeño de los modelos de programación y reprogramación de la producción propuestos en esta investigación, se compararon con otros modelos similares encontrados en la literatura, evidenciando que el modelo de programación obtuvo resultados similares o iguales a éstos en tiempos computacionales cortos. Y luego, a partir de la solución anterior, usando el modelo de reprogramación se simuló aleatoriamente la interrupción de alguno de sus trabajos obteniendo una nueva secuencia con características similares a la inicial. Además, se realizó un caso de estudio para una fábrica de calzado con un horizonte de programación de una semana, obteniendo con el modelo de reprogramación nuevas secuencias que minimizan la diferencia entre los tiempos de inicio y la tardanza total con respecto a la programación inicial, y a su vez, obteniendo resultados muy similares al Makespan y la Tardanza de la programación inicial.

3. DESCRIPCIÓN DEL PROYECTO

3.1. Planteamiento del Problema

Entre los factores que más afectan las actividades económicas en Colombia, los que tienen mayores incidencias son: la corrupción, los impuestos, ineficiencia del gobierno e infraestructura inadecuada (World Economic Forum, 2017).

En cuanto al índice de desempeño logístico (Logistic Performance Index – LPI) evaluado por THE WOLRD BANK, Colombia se encuentra en el puesto 58 con un valor de 2.94, mientras que Alemania con el mejor puesto a nivel mundial tiene un valor de 4.2 y Chile por su parte, el mejor de la región, está ubicado en el puesto 34 con un valor de 3.32 (Cabe anotar que la última actualización del indicador fue desarrollada en el año 2018). En la figura 1 se puede observar que Colombia tiene un desempeño logístico inferior a la media de los encontrado en

países desarrollados, como por ejemplo Alemania, y también con respecto a algunos países de la región como Chile (THE WORLD BANK, 2018).

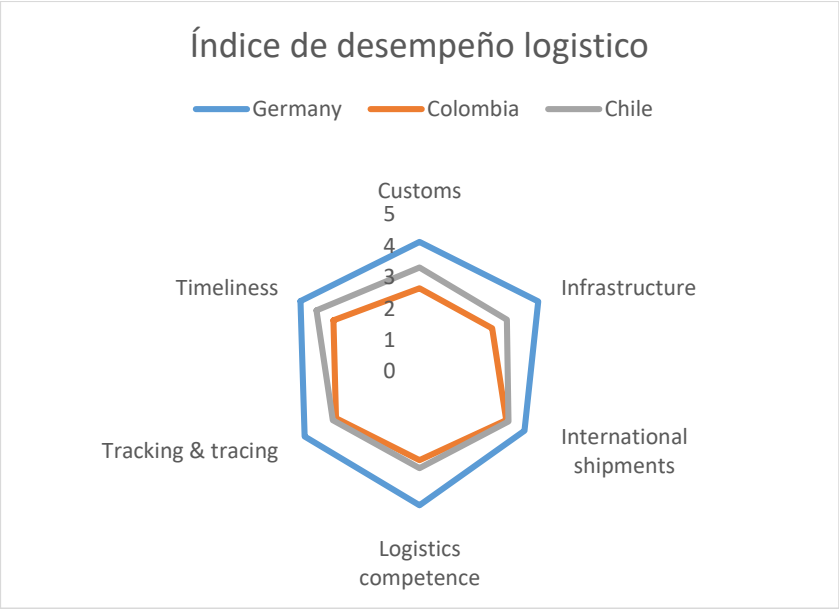

Figura 1: Índice de desempeño logístico de Alemania, Chile y Colombia

Fuente: World Economic Forum

El costo de servicios logísticos, por ejemplo, de importar mercancía corresponde a un 36,6% correspondientes a las categorías de transporte internacional (4,6%); derechos aduaneros (14,8%), y costos internos no arancelarios (17,2%). Estas dos últimas categorías muestran que los costos más altos se generan dentro de la logística interna de nuestro país, siendo un obstáculo para el ingreso de las mercancías. Como consecuencia, se puede inferir que Colombia tiene desventajas en cuanto a la logística de las importaciones dificultando su competitividad frente a otros países.

Uno de los elementos que tiene mayor influencia en el LPI es el estado de las carreteras en Colombia, las cuales no están en las mejores condiciones. Esta situación genera retrasos en la entrega de productos afectando en las empresas sus planes de producción y comercialización. Actualmente el pais tiene pavimentado el 88,73% de la red vial primaria, y de esta solo el 45,34% (Figura 2) está en buen estado (INSTITUTO NACIONAL DE VIAS, 2019). Por lo anterior,

Colombia presenta muchas oportunidades de mejora que requieren de políticas que le permitan subir su nivel en cuanto al Índice de desempeño logístico logrando así mayor confiabilidad en la gestión logística que necesitan las empresas.

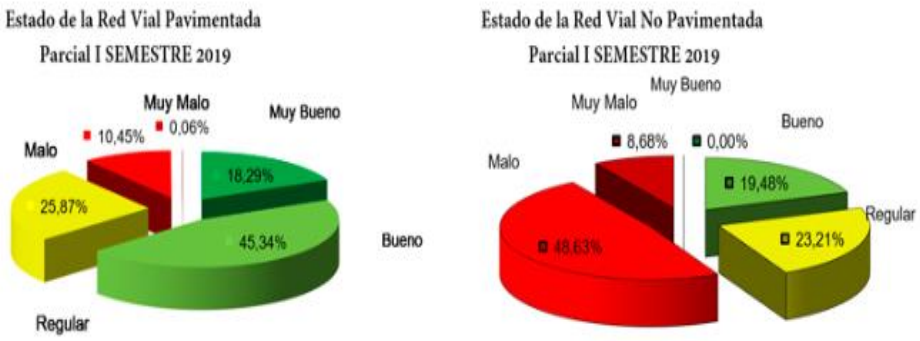

Figura 2: Estado de la Red vial colombiana.

Fuente: Instituto nacional de vías

La falta de herramientas formales de planeación hace que las empresas no consideren perturbaciones en el sistema de producción como producto de eventos generados en los procesos logísticos. Ejemplos de estas situaciones son: retraso en la llegada, problemas de calidad o la no disponibilidad / escasez de materias primas. En el análisis de sensibilidad de los planes de producción, debe entonces considerarse esquemas de planeación, programación y, en entornos con alta incertidumbre en procesos logísticos, reprogramación de la producción con el objetivo de minimizar los efectos de estas perturbaciones en el desempeño del sistema (Vieira, Herrmann, & Lin, 2003).

Los procesos de programación de la producción se conocen por ser problemas complejos debido a la incertidumbre que presenta y al tamaño de las instancias en entornos reales, lo cual lo hace NP-Hard debido a la naturaleza combinatoria del problema de optimización asociado (Luh & Feng, 2003). Abordar la solución de estos modelos mediante el uso de heurísticas y metaheurísticas permite generar funciones objetivo que consideren el efecto de fluctuaciones en los tiempos del proceso, de transporte y de entrega final. Además se pueden

considerar el efecto en los tiempos de alistamiento y en los costos totales. Este tipo de problemas son en ocasiones complejos de solucionar por métodos exactos en un tiempo computacional razonable (Acevedo Chedid, Salas Navarro, Ospina Mateus, & Santander-Mercado, 2017).

Sin embargo, existen muchos de los elementos que afectan al sistema y que son difíciles de anticipar. Es por esto que se deben tener estrategias de mitigación de sus efectos, como lo son los esquemas de reprogramación, los cuales permiten actualizar un programa productivo existente como respuesta ante las interrupciones del sistema (Vieira, Herrmann, & Lin, 2003). Desde el punto de vista de la reprogramación se debería asegurar que no se presente una nueva solución que sea completamente diferente a la inicialmente planteada, debido a que la configuración del sistema responde a una planeación a mediano plazo que debe, en lo posible, respetarse. Esta situación genera que las funciones objetivo a trabajar en el problema de optimización conexo, deben apuntarle a la minimización de la diferencia entre la solución inicial y la reprogramada, conservando buenos niveles de desempeño en cuanto a las métricas del problema.

Teniendo en cuenta lo anterior, se tiene como propósito responder a la siguiente pregunta de investigación: ¿Cómo, mediante el diseño de un modelo para reprogramación de la producción en ambientes Flexible Flow Shop bajo condiciones de incertidumbre en el proceso de aprovisionamiento de materias primas, se puede mejorar la confiabilidad sobre las métricas de Makespan y de tardanza del programa de producción?

En el marco de esta investigación, se entiende el término ‘confiabilidad’ como una medida del nivel de confianza en cuanto a minimizar desviaciones de la programación de la producción, a pesar de perturbaciones futuras en su desarrollo. Es decir, que se espera que aún en presencia de eventos que generen perturbaciones, las nuevas metricas de makespan y de tardanza sean lo más cercanas posible a sus respectivas contrapartes del plan original, luego de la reprogramación de la producción.

3.2. Justificación

En la presente sección se realiza la exposición de la justificación acerca de la relevancia de la investigación, desde la perspectiva teórica y práctica.

3.2.1. Teórica

La reprogramación de la producción es un proceso que pocas veces se realiza de manera formal en las organizaciones, ya que muchas de las herramientas computacionales están diseñadas para la programación de operaciones, pero no para la reprogramación. Esta situación genera que se corran nuevamente algoritmos de programación para encontrar la "nueva solución óptima", pero obviando la planeación logística alrededor del programa anterior. Esta situación hace necesario el desarrollo de protocolos de reprogramación de la producción que permitan reaccionar de manera efectiva ante eventos e interrupciones inesperadas (por ejemplo: cambios en la liberación de los trabajos, llegada y cancelación de pedidos, pedidos urgentes, demoras en los tiempos de ejecución de los procesos, averías de máquinas, problemas de calidad, entre otros) manteniendo en gran medida la planeación logística relacionada. Como resultado de la reprogramación ante estas fuentes de incertidumbre, se espera un nuevo programa rentable y con las mínimas variaciones con respecto al plan inicial (Hall & Potts, 2010).

Los investigadores han abordado el problema de los modelos de reprogramación para cerrar la brecha entre los modelos teóricos y los problemas reales en la industria. Además, se hace necesario debido a que los problemas de producción se vuelven cada vez más complejos debido a la necesidad de tener portafolios más amplios de producto, que implican mayor cantidad de alistamientos, diferentes materiales, curvas de aprendizaje, capacitación entre otros aspectos. Estos efectos se ven amplificados debido a la dinámica de los mercados en donde se tienen cambios en los hábitos de consumo de manera más frecuente (ciclos de vida más cortos) e influenciado por tendencias mundiales.

Antes del diseño de modelos de reprogramación se deben tener en cuenta algunas consideraciones y decisiones de acuerdo a las características de cada problema de producción (García, Marquez, & Burtseva, 2015): ¿Cuáles eventos deben ser atendidos?, ¿cuál evento debe ser prioridad? y ¿cuál debería ser la frecuencia de la reprogramación? Uno de los desafíos en este tipo de problemas es evaluar el impacto de las estrategias de reprogramación en la mejora de la robustez del programa y su estabilidad.

El problema de la reprogramación de la producción, al igual que el de programación, presenta restricciones tecnológicas y de la planeación del proceso global que determinan la secuencia de realización de los trabajos. Por otro lado, la diferencia entre el problema de la reprogramación y la programación inicial está en el inicio de la ejecución de las operaciones según el cronograma. El objetivo del problema de la reprogramación es minimizar la desviación utilizando dos medidas de rendimiento: Medida de eficiencia (Makespan) y medida de estabilidad, como se entiende la desviación con respecto al programa inicial (Abumaizar & Svestka, 1997).

El problema de programación de la producción es a menudo NP Hard (Church & Uzsoy, 1992) para el caso de una sola máquina. La solución de esta clase de problemas se vuelve aún más compleja cuando está en medio de un ambiente dinámico. Farn & Muhlemann (1979), por ejemplo, propusieron un modelo llamado política hibrida de reprogramación que reaccionaba frente a eventos como nuevos trabajos, trabajos urgentes, averías de máquinas, entre otros. Este modelo permite decidir cuándo es conveniente realizar la reprogramación teniendo algunos criterios denominados excepciones utilizando como métrica minimizar la tardanza máxima (T_{max}), con lo cual se prioriza cuando es necesario realizar una reprogramación

Para el caso de problemas de máquinas en paralelo, el tiempo final del proceso, generalizado como $R\,||\Sigma Ci$, son los únicos problemas no preventivos con tiempos de procesos arbitrarios que se resuelven de forma polinomial. Por otro lado, para el caso de problemas de Makespan y tiempo de flujo ponderado total son NP-Hard (Brucker, 2006). Se ha comprobado que los problemas de tipo Job Shop y Flow Shop son NP-Hard debido a dos razones (GAREY, JOHNSON,

& SETHI, 1976): 1) son computacionalmente intratables, 2) No se pueden resolver con un algoritmo eficiente.

En general, la programación de la producción es un proceso complejo por la estructura y componentes que presenta y a su tamaño, lo cual lo hace NP-Hard debido a la naturaleza combinatoria del problema de optimización asociado (Luh & Feng, 2003). Es por esto que se deben tener estrategias de mitigación de sus efectos, como lo son los esquemas de reprogramación, los cuales permiten actualizar los procesos de un programa productivo existente como respuesta ante las interrupciones del sistema (Vieira et al., 2003).

3.2.2. Práctica

El principal objetivo de cualquier empresa es mejorar su rentabilidad y competitividad por lo que una buena optimización en la asignación de los recursos le permitirá acercarse a este objetivo. (Salido, Escamilla, Barber, & Giret, 2016)

En la programación de la producción se busca asignar recursos de manera óptima en un periodo de tiempo determinado para cumplir con la planeación de la producción. Sin embargo debido a las fuentes de variabilidad e incertidumbre que afectan el plan de producción, se han realizado sistemas de programación robustos donde consideran esta incertidumbre de manera que se pueda obtener un plan de producción más acorde a la realidad (Bougeret, Alves Pessoa, & Poss, 2018).

En un proceso productivo las materias primas son recursos críticos, por lo que las compañias utilizan generalmente tiempos de esperas seguros e inventarios de seguridad que le permitan mantener el flujo en su proceso productivo. Sin embargo, aunque las compañias puedan tener en cuenta estas variables según la materia prima y el proveedor, la recepcion de éstas pueden ser tardías ocasionando retrasos productivos en el proceso (Vidal & Goetschalckx, 2000).

Teniendo en cuenta lo anterior, se refleja claramente que existen eventos o interrupciones que pueden ocurrir durante el desarrollo de los procesos en una empresa de manufactura afectando la consecución de sus objetivos. Debido a que estos eventos son dificiles de anticipar la reprogramacion de la produccion es necesaria para mitigar sus efectos adversos ya que se obtiene una nueva programación o actulizacion del programa actual.(Salido et al., 2016).

3.3. Marco teórico

La reprogramación de la producción busca establecer un programa que determine el orden de la realización de todas las operaciones de un sistema en un tiempo determinado dado que ocurrió alguna interrupción durante el proceso. Este programa debería ser el óptimo para el sistema pueda cumplir con la planeación de la producción inicial a través de una serie de decisiones teniendo en cuenta las restricciones del sistema, sin perder de vista que se pueden presentar imprevistos. Se espera que la nueva programación que no sea completamente diferente a la inicial para que se mantengan las estimaciones de rentabilidad y se logre la sostenibilidad en el desarrollo de la operación (Abumaizar & Svestka, 1997). De acuerdo a esto la reprogramación de la producción se realiza en las siguientes etapas (Yamamoto, 1985): 1) Planeación y programación de la producción, 2) Control de la producción, y 3) Reprogramación de la producción.

3.3.1. Planeación y programación de la producción

De acuerdo con la lógica presentada anteriormente se definen cada una de las etapas para la planeación y programación de la producción.

- **Planeación agregada de la producción**

En la planeación de la producción se utiliza la técnica de Planeación Agregada que nos permite garantizar la demanda futura teniendo en cuenta la capacidad de la planta según los equipos disponibles, personal y materiales donde generalmente se maximizan beneficios o disminuyen costos (H Leung, Wu, & Lei, 2003)**.** En la literatura, los principales objetivos de una planeación

agregada típica de la producción son (Leung & Wu, 2004): 1) minimizar costos de producción; 2) minimizar costos de mano de obra; 3) minimizar nivel de inventario; y 4) mantener una mano de obra equilibrada.

Los métodos para resolver el problema de planificación agregada se pueden clasificar en 3 grupos (Boiteux, Corominas, & Lus, 2007): Métodos de comparación de alternativas, se utiliza para comparar alternativas según varios escenarios y criterios, es muy útil para evaluar estrategias; métodos con reglas de decisión que utilizan expresiones matemáticas lineales o no, que incluyen información del sistema (demanda, nivel de stock, costos, entre otros) para generar el plan agregado; y modelos de programación lineal que por su característica de modelación libre se puede incluir todas las restricciones que se requieran para obtener una planeación agregada más cercana a la realidad.

Para realizar la planificación agregada de la producción se debe en cuenta las variables de decisión del problema, los parámetros y constantes, y las restricciones del sistema (Boiteux, Corominas, & Lus, 2007). En cuanto a las variables de decisión se tienen:

- Nivel de inventario: Cantidad de inventario de los productos en cada periodo
- Nivel de fuerza laboral: Cantidad de personas que se contratan o que se despiden.
- Nivel de producción: Cantidad de unidades a realizar de cada producto
- Subcontratación: Cantidad de productos realizados bajo la modalidad de subcontrato
- Horas extras: cantidad de horas extras utilizadas para la realización de los productos

Regularmente los parámetros del sistema son:

- Demanda: Pronostico de demanda de cada producto.
- Costo de producción: Costo de producir una unidad de cada producto, puede ser en tiempo regular o en tiempo extra.
- Costo de inventario: Costo de mantener una unidad de cada producto.

- Costo de faltantes: Costo por cada unidad de producto no disponible para suplir la demanda.
- Costo de contratar o despedir: Costo de contratar o despedir una persona para cada periodo.
- Inventario inicial: Cantidad de inventario inicial.
- Nivel de fuerza laboral inicial: Cantidad de trabajadores al iniciar la producción.
- Tiempo en mano de obra o de máquina del producto: Tiempo necesario para la elaboración de un producto.
- Horas de trabajo disponibles: Horas hombre disponibles en cada periodo, pueden ser en tiempo regular o extra.

Las restricciones generales del sistema son:

- Especificaciones del producto: características físicas, químicas y de calidad del producto.
- Restricciones técnicas: capacidad de la maquinaria, mano de obra, presupuesto, entre otros.
- Requerimientos del mercado: respecto a las fechas de entrega y a la necesidad de evitar la ruptura de stock.
- Aspectos de operación, relacionados con el problema del flujo y almacenamiento según la distribución de planta de la fábrica, limitación en uso de horas extras y subcontrataciones, relacionadas al nivel de fuerza laboral (no se puede contratar y despedir en el mismo periodo) entre otros.

- **Plan maestro de la producción y MRP**

El Plan Maestro de Producción (MPS) procede del Plan Agregado de Producción y representa la estrategia con la que las empresas pretenden satisfacer la demanda de manera que decide qué, cuánto y cuándo se realizarán los productos a mediano plazo (Khiong Yang & Jacobs, 1999). Del MPS se desprende el plan de requerimiento de materiales (MRP) que es una técnica que desglosa

los materiales requeridos para la elaboración de los productos en cantidades necesarias y fechas exactas para cumplir con el MPS.(N. Mustafizul & S. Mainul, 2002). Luego que el MRP se ha generado, se realiza la programación inicial de la producción que organiza el trabajo de modo que se cumpla con los objetivos planteados por la organización (Babtiste & Favrel, 1993).

- **Programación de la producción**

La planificación de los procesos y las actividades en la programación de la producción generalmente se organiza de manera secuencial en un entorno estático (Wong , Leung, Mak, & Fung, 2006) cuando en la realidad la producción está envuelta en un ambiente dinámico. Es inevitable que ocurran eventos inesperados en la ejecución de la producción como averías de máquinas, retraso de materias primas, problemas de calidad, pedidos urgentes, cancelación de pedidos, retraso en los tiempos de procesos, indisponibilidad o ausencia de trabajadores, entre otros, por lo que un programa de producción puede volverse ineficiente e inviable debido a los cambios.

La programación de la producción puede ser predictiva o reactiva (Babtiste & Favrel, 1993). En el caso de la programación predictiva se realiza una programación simple con ayuda de un algoritmo de investigación de operaciones, la simulación de un sistema, un sistema de soporte de decisiones, entre otros. Esta programación usualmente arroja un diagrama de Gantt donde se asignan las operaciones a las máquinas en un tiempo determinado. La programación reactiva permite generar una nueva programación en respuesta a eventos inesperados, también se le conoce como reprogramación (Wong , Leung, Mak, & Fung, 2006).

- **Tipos de problema de programación y reprogramación de la producción**

  - Problemas de una sola máquina (Single Machine): Hace referencia al problema de programación de trabajos con sus correspondientes tiempos de procesos en una sola

máquina (Lazo Eche, Gutierrez Segura, & Vergara Moreno, 2016). Estos trabajos pueden tener fechas establecidas de entrega y penalizaciones por entregas tardías de los mismos.

Los primeros resultados de este tipo de problemas surgieron luego de los trabajos de Jhonson, lo que permitió que se desarrollaran algoritmos basados en reglas de prioridad (Potts & Strusevich, 2009). Entre estos algoritmos se encuentran: SPT desarrollado por Smith en 1956 donde se ordenan los trabajos según el orden creciente de sus tiempos de proceso, EDD desarrollado por Jackson en 1955 en donde se ordenan los trabajos se ordenan según la fecha de entrega más cercana, LPT desarrollada por Baker en 1974 en la cual se organizan los trabajos según los tiempos de procesos más largos, entre otros.

Para la solución de los problemas de programación de la producción, los algoritmos con reglas de prioridad se han utilizado por varias décadas debido a que son prácticos, fáciles de aplicar y ofrecen buenas soluciones. El inconveniente o defecto de estas reglas de prioridad es que están limitadas a una solo función objetivo. En recientes investigaciones se propone un modelo para single machine con DTS ("Decision Theory"approach for scheduling / sequencing) para una gran variedad de objetivos (Gahm, Kanet, & Tuma, 2019), donde los resultados arrojaron que DTS es una alternativa flexible y viable a los enfoques de reglas de prioridad.

- Problemas de máquinas en paralelo: El problema de programación de máquinas paralelas consiste en *m*-máquinas, generalmente idénticas (se espera que no se tengan diferencias significativas en los tiempos de procesamiento, ni en las métricas de calidad), ubicadas en paralelo para *n* trabajos, pero cada trabajo se puede programar solo en una sola máquina (Salazar Hornig & Medina S, 2013). Una de las reglas más utilizadas para resolver este problema es LPT (Largest Processing Time), o mediante esquemas de programación lineal dado que en su primera etapa puede verse como un problema de asignación si la métrica de evaluación es el Makespan.

En investigaciones recientes también se ha resuelto este tipo de problemas con el uso de programación dinámica. Por ejemplo, para el problema de máquinas paralelas permitiendo el rechazo de trabajos se presentó un nuevo modelo de reprogramación con un enfoque basado en programación dinámica y de ramificación y precio. Este tipo de problema es Np Hard en el sentido ordinario cuando el número de máquinas es fijo (Wang, Yin, & Cheng, , 2018).

- Talleres de producción continua (Flow Shop): El problema de programación en entornos de tipo Flow Shop fue desarrollado por Johnson en 1954. Este problema en general consiste en un set *N* de trabajos y con *m*-máquinas sucesivas, donde cada trabajo tiene una operación que se realiza en *m* con un tiempo de proceso comúnmente denominado como $p_{ij.}$ En este caso todos los trabajos tienen el mismo orden o ruta de procesamiento (Potts & Strusevich, 2009).

  Para resolver este tipo de problema se han propuesto modelos de reprogramación con el uso de algoritmos genéticos, en el caso de múltiples líneas de producción para Flow Shop, se han presentado un modelo para la reprogramación donde se utilizan los tiempos sobre asignados de las actividades para utilizarlos en caso de que eventos inesperados, el solucionador de este modelo está basado en el algoritmo genético.

- Talleres de Producción Intermitente (Job Shop): Este tipo de problema consiste en una serie de máquinas donde los trabajos tienen diferentes secuencias o rutas de procesamiento dentro del set de *m*-máquinas. Algunos de los algoritmos utilizados para resolver este tipo de problema son reglas de despacho, técnicas basadas en el análisis del cuello de botella y heurísticas locales como búsqueda tabú, algoritmos genéticos, recocido simulado entre otros (Lee, Piramuthu, & Tsai, 1997)

  El uso de programación dinámica para resolver este tipo de problemas tiene mayores ventajas con respecto a la programación robusta estocástica como trazabilidad, fácil

implementación y conocimiento del problema. Una reciente investigación demostró el beneficio de esta reprogramación sobre métodos estáticos, con presencia de incertidumbre y aspectos dinámicos del problema (Larsen & Pranzo, 2019).

- Flexible Flow Shops: El problema de Flexible Flow Shop (FFS) fue identificado en los 70s y ha sido objeto de investigación en las últimas décadas (Choi & Wang, 2012). El FFS consta de una serie de etapas de producción, los trabajos deben ir hacia cada etapa en el mismo orden y cada etapa tiene varias máquinas, generalmente idénticas en paralelo.

  Uno de los algoritmos utilizados en modelos de reprogramación para resolver este problema es el de búsqueda en vecindario variable (Variable Neighborhood Search, VNS). En un estudio reciente se propuso un modelo basado en este algoritmo para resolver el problema de reprogramación FFS ante alguna interrupción en el sistema, donde la función objetivo es minimizar la tardanza ponderada total, los resultados de este modelo fueron eficientes comparados con 3 algoritmos de reglas de despacho y otro eficiente modelo de reprogramación (Rahmani & Ramezanian, 2016).

- Flexible Job Shop: El problema de Flexible Job Shop (FJS) es una generalización del problema tipo Job Shop (JS) clásico, donde a diferencia del problema JS, las operaciones pueden procesarse en cualquiera de un conjunto de máquinas disponibles en cada etapa de su proceso, por lo que su complejidad es mayor. Por lo tanto, se hace necesario definir las rutas de los trabajos, se debe definir cuál operación se realiza en cada máquina disponible (Pezzella, Morganti, & Ciaschetti, 2008).

  El problema de FJS debido a su complejidad es NP-Hard completo, en la literatura de reprogramación, los autores proponen muchas técnicas y enfoques que abordan el problema de las incertidumbres en los problemas de programación. En un estudio reciente se propone un modelo basado en la heurística PSO (Enjambre de partículas), que

tiene como objetivo minimizar el Makespan después de una interrupción en el proceso por avería de máquinas (Nouiri, Bekrar, Jemai, & Ammari, 2018).

3.3.2. Control de la producción

En el mundo competitivo actual, la satisfacción al cliente es sin duda uno de los factores en el cual las empresas colocan su mayor esfuerzo para mantenerse a la vanguardia en el mercado. Cumplir con las expectativas del cliente incluye ofrecerle una gran variedad de productos, plazos de entrega cortos y tiempos de entrega confiables. Por lo cual, la importancia de un sistema de control de la producción efectivo y eficaz es vital para la sostenibilidad de cualquier empresa dedicada a la manufactura (Jaegler, Jaegler, Burlat, & Lamouri, 2018). A continuación, se presentarán algunas metodologías para el control de la producción:

- **Control de la actividad de la producción CAP**

El control de la actividad de la producción (CAP) vigila la actividad real en la fabricación de un producto según la planificación previa y el orden de los trabajos establecidos en este. El CAP refleja varias informaciones sobre la ejecución del plan de producción como: pedidos recién liberados, estado de los pedidos existentes, información de la ruta de los trabajos, información del tiempo de espera, estado de los recursos, información referente al personal, herramientas, entre otros (Chapman, 2006).

- **Planeación Jerárquica de la producción**

El control de la producción se refiere a la coordinación de las actividades de producción en un horizonte de tiempo determinado, debido a que esta coordinación es compleja a menudo se dividen las actividades de producción en niveles altos y bajos. Uno de los sistemas de control de producción utilizados es la planeación de producción jerárquica (Hierarchical Production Planning (HPP)), la cual es una estructura de planificación y control organizada jerárquicamente para reducir la complejidad del problema del control (McKay & Wiers, 2003).

- **KANBAN**

Este sistema fue desarrollado por Toyota Motor Corporation y consiste en tener un registro visible, puede ser a través de una tarjeta, tablero o una señal visual o electrónica, que comunica el nivel de inventario entre las estaciones de trabajo. De esta forma el sistema minimiza el inventario y entrega la materia prima o los productos semielaborados a la siguiente máquina sólo cuando ésta lo solicite (Sharma & Singla, 2019).

El sistema Kanban mantiene activo el proceso de reabastecimiento en una cadena de producción que está formada por centros de trabajo donde los materiales y la información circulan de principio a fin, es decir desde que ingresa la materia prima al sistema hasta que se entrega el producto terminado al cliente, por lo cual, para mantener el control del proceso, Kanban está basado en una serie de principios (Lendínez, 2019):

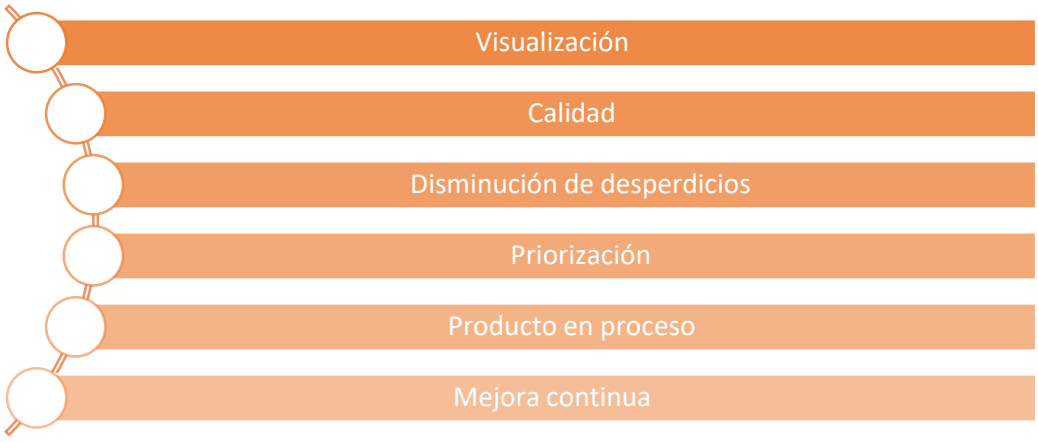

Figura 3: Principios de Kanban

- **CONWIP**

Es una forma generalizada de Kanban que trabaja con producto en proceso que se utiliza principalmente en las líneas de producción, al igual que Kanban se basa en señales para revisar el inventario dentro del sistema. En general, el objetivo principal de CONWIP es controlar la

cantidad total de trabajo en el entorno, manteniendo constante los niveles de inventario (Jaegler, Jaegler, Burlat, & Lamouri, 2018).

Se puede considerar a CONWIP como un sistema hibrido entre los sistemas tipo Pull y Push, donde se controla la cantidad de materiales que entran al sistema de manera que si el sistema se llena no se le da paso a ningún otro trabajo hasta que no se produzca la salida de otro, de esta manera se garantiza un flujo constante de los materiales dentro del sistema (Spearman, Woodruff, & Hopp, 1990).

- **Control de la producción con el uso de sensores**

Una de las herramientas para supervisar y controlar el proceso de producción es el uso de sensores que suministran la información necesaria del sistema productivo. Estos sensores permiten adquirir información sobre las variables críticas del proceso y el monitoreo de estos, de esta forma se pueden tomar decisiones que permiten eliminar o minimizar el impacto de algún evento inesperado en la planeación de la producción.

Con las grandes cantidades de datos obtenidas con el uso de sensores se han construido modelos predictivos llamados sensores de software, el término de software se debe a que son modelos realizados con programas informáticos y sensores ya que los modelos proporcionan información similar a la de sus contrapartes de hardware. Los sensores de software se utilizan con el fin de monitorear los procesos para detectar y diagnosticar fallas en el proceso productivo y existen dos clases de sensores de software los basados en modelos y los basados en datos (Kadlec, Gabrys, & Strandt, 2009).

- **Control de la producción en la industria 4.0**

Los sistemas de control de producción tienen como objetivo asegurar el control de la fabricación de los productos de una empresa de acuerdo con el plan de producción establecido. En un sistema de producción es necesario considerar la variabilidad del mercado, la gran variedad de productos y la exigencia en tiempos de entrega como de vital importancia para mantenerse

competitivos en el mercado (Wang, Jiang, & Lu, 2018). Frente a esta problemática La industria 4.0 se muestra como una alternativa actual que pretende interconectar todas las partes de una empresa para lograr una automatización efectiva e inteligente.

3.3.3. Reprogramación de la producción

La programación de la producción ayuda a los líderes del proceso a organizar y coordinar las actividades de manera que los recursos sean utilizados eficientemente teniendo en cuenta las especificaciones del producto y la satisfacción del cliente. La programación de la producción permite identificar conflictos en los recursos, controlar la liberación de los trabajos y asegurar que la materia prima sea pedida a tiempo. Sin embargo, debido a que los sistemas de producción son dinámicos y pueden ocurrir eventos inesperados se hace necesario construir esquemas de reprogramación que permitan obtener un nuevo programa rentable y con las mínimas variaciones con respecto al plan inicial (Hall & Potts, Rescheduling for job unavailability, 2010). Algunas de estos eventos inesperados que pueden generar perturbación en un plan de producción son:

- Averías de máquinas. (Muhlemann, Lockett, & Farn, 1982) (Yamamoto, 1985) (Wang, Yin, & Cheng, , 2018)
- Llegada y/o cancelación de trabajos. (Church & Uzsoy, 1992) (Bierwirth & Mattfeld, 1999) (Sun & Xue, 2001)
- Tiempo de proceso. (Daniels & Kouvelis, Robust scheduling to hedge against processing time uncertainty in single-stage production, 1995) (Larsen & Pranzo, 2019)
- Pedidos urgentes. (Jain & Elmaraghy, 1997) (Arnaout J. P., 2014)
- Retraso en la llegada o escasez de materiales. (Li, Shyu, & Adiga, 1993) (Fang & Xi, 1997)
- Reproceso o problemas de calidad. (Li, Shyu, & Adiga, 1993) (Li, Li, Li, & Hu, 2000)

- **Terminología utilizada en Reprogramación de la producción**

A continuación, se consideran una serie de términos relacionados con reprogramación de la producción según Viera et al (2003):

- Sistema de manufactura: Permite la organización de equipo, personal, y la información concerniente a la fabricación de los productos.
- Liberación de pedidos: Controla la entrada de los trabajos al sistema productivo teniendo en cuenta cuando deberían ser ingresados al sistema.
- Control de planta: Determina la asignación de las operaciones teniendo en cuenta las personas y los equipos que se deben utilizar para realizarlas y cuando debería hacerse.
- Programación de la producción: Especifica el inicio y el final de cada trabajo según cada recurso de la producción asignado a éste.
- Reprogramación de la producción: Es el proceso de actualizar la programación de la producción como respuesta a una interrupción o cambio en la producción.
- Política de reprogramación: Especifica cuando y como se debe realizar la reprogramación. En la política se debe establecer cuales eventos son los desencadenantes para realizar la reprogramación, estos eventos pueden ser predecibles o impredecibles.

- **Métricas de desempeño**

Viera et al (2003) identificaron en la literatura las métricas más comunes para evaluar el desempeño de sistemas de programación y reprogramación:

- Eficiencia de la programación: Makespan, tardiness, flow time, utilización de recursos y tardanza máxima.
- Estabilidad de la programación: Desviación del tiempo de inicio entre la nueva programación y la inicial, media de la diferencia de secuencia entre la nueva programación y la inicial, medidas de desempeño basadas en tiempo.
- Métricas basadas en costos: Minimizar costos por trabajos que comienzan antes del tiempo esperado, minimizar costos de producto en proceso, utilidad, minimización de costos totales, minimizar costos computacionales, minimizar costos de configuración (alistamientos) y minimizar costos de transporte.

3.4. Estado del arte

Una de las primeras investigaciones sobre reprogramación de la producción fue realizada por Farn y Muhlemann (1979), donde se abordó el problema reprogramación de una sola máquina en condiciones de incertidumbre. En esta investigación las órdenes de trabajo llegan dinámicamente y se debe tomar una decisión en cuanto a la frecuencia de la reprogramación. En el análisis de esta investigación se evaluó de forma estática y dinámica concluyendo que la reprogramación dinámica es la mejor opción debido a que tiene menor tiempo de ejecución.

En la literatura sobre programación y reprogramación de la producción se encuentran diferentes tipos de incertidumbre como averías de máquinas, cancelación y llegada de trabajos, pedidos urgentes, problemas de calidad y reproceso, ausencia de trabajadores, tiempos de proceso, lead time, entre otros, que pueden afectar la consecución de los objetivos de la planificación de la producción. Por lo tanto, se han realizado sistemas de programación robustos donde consideran esta incertidumbre de manera que se pueda obtener un plan de producción más acorde a la realidad y en caso de la reprogramación se debería asegurar que esta no presente una nueva solución que sea completamente diferente a la inicialmente planteada (Bougeret, Alves Pessoa, & Poss, 2018).

A continuación, se presenta el resultado de la búsqueda de diferentes artículos que abordan el problema de programación y reprogramación de la producción cuando suceden eventos inesperados:

Authors	Scheduling				Uncertainty																
	Scheduling	Rescheduling	Static	Dynamic	Lead time	Demand	Incorrect work	Quality problems	Process time	Change of production order (set up)	Affected operation	Arrival and / or cancellation of work	Job release	Re-entring lines (jobs come back to a machine)	Urgent Job Arrivals	Unavailable resource	Sensitive jobs	Sickness of workers	Due dates	Machine Breakdown	Government policy	Interruption in the sequence of work
Farn, C. K., & Muhlemann, A. P. (1979)		X		X																		
Muhlemann, A. P., Lockett, A. G., & Farn, C. K. (1982).		X		X					X											X		
YAMAMOTO & NOF (1985)	X	X	X																	X		
Christy, D. P., & Kanet, J. J. (1988)		X				X																
Bean, J. C., Birge, J. R., Mittenthal, J., & Noon, C. E. (1991)		X		X																X		
Church & Uzsoy (1992)		X	X									X										
Chang, S. C., & Hsieh, F. S. (1992)	X	X	X									X								X		
BAPTISTE, P., & FAVREL, J. (1993).		X	X																	X		
Wu, Storer &Pei-Chan (1993)		X		X																X		
Li, R. K., Shyu, Y. T., & Adiga, S. (1993)		X		X							X									X		
Zweben, Davis, Daun, and Deale (1993)	X	X																				
Kumar, P. R. (1994).	X			X								X		X								
Jorge Leon, V., David Wu, S., & Storer, R. H. (1994)	X			X																X		
Kim, M. H., & Kim, Y. D. (1994)	X	X		X																		
Daniels, R. L., & Kouvelis, P. (1995)	X			X					X													

Wu, H. H., & Li, R. K. (1995)		X		X							X											
Unal, A. T., Uzsoy, R., & Kiran, A. S. (1997)		X		X								X										
Fang, J., & Xi, Y. (1997)		X		X								X							X	X		
Daniels, R. L., & Carrillo, J. E. (1997)	X			X					X													
Chang, J. W., & Luh, Y. P. (1997)	X	X	X												X							
JAIN, A. K., & ELMARAGHY, H (1997)	X	X		X								X			X					X		
Abumaizar, R. J., & Svestka, J. A. (1997)		X		X																X		
Byeon, E. S., Wu, S. D., & Storer, R. H. (1998)	X		X																			
Mehta, S. V., & Uzsoy, R. M. (1998)		X		X																X		
Guo, B., & Nonaka, Y. (1999)		X	X																	X		
O'Donovan, R., Uzsoy, R., & McKay, K. N. (1999).		X		X													X			X		
Akturk, M. S., & Gorgulu, E. (1999)		X		X																X		
Bierwirth, C., & Mattfeld, D. C. (1999	X	X	X	X								X										
Li, H., Li, Z., Li, L. X., & Hu, B. (2000)		X					X	X							X					X		
Viera, Herrman & Lin (2000)		X		X						X												
Vieira, G. E., Herrmann, J. W., & Lin, E. (2000).		X	X	X								X								X		
Sabuncuoglu, I., & Bayız, M. (2000)		X	X																	X		
Sun, J., & Xue, D. (2001).	X			X								X						X		X		
Cowling, P., & Johansson, M. (2002)		X		X					X													
Dolgui, A., & Ould-Louly, M. A. (2002)	X			X	X																	
Dupon, A., Van Nieuwenhuyse, I., & Vandaele, N. (2002).	X		X		X										X							X
Honghong, Y., & Zhiming, W. (2003)		X		X								X			X					X		
Bidot, J., Laborie, P., Beck, J. C., & Vidal, T. (2003)		X		X					X													

Bajestani, M. A., & Beck, J. C. (2011)		X		X								X										
Al-Hinai, N., & ElMekkawy, T. Y. (2011).		X																		X		
Dong, Y. H., & Jang, J. (2012).		X		X																X		
Yu, S. P., & Pan, Q. K. (2012)		X		X																X		
Choi, S. H., & Wang, K. (2012)	X			X					X													
Katragjini, K., Vallada, E., & Ruiz, R. (2013).		X		X					X			X								X		
Lanza, G., Stricker, N., & Moser, R. (2013, December)		X		X																X		
Zhang, L., Gao, L., & Li, X. (2013).		X		X								X								X		
Liu, Z., & Ro, Y. K. (2014)		X		X																X		
Shirai, K., & Amano, Y. (2014)		X		X	X																	
Arnaout, J. P. (2014)		X	X												X					X		
Lv, S., & Qiao, L. (2014).		X		X								X								X		
Li, J. Q., Pan, Q. K., & Mao, K. (2015)		X		X					X			X	X							X		
Zhang, J., & Qiao, C. (2015, April).	X			X					X													
Shen, X. N., & Yao, X. (2015)		X		X								X			X					X		
Gürel, S., & Cincioğlu, D. (2015)		X		X																X		
Rahmani, D., & Ramezanian, R. (2016).		X		X								X										
Zhenyu, G., Linfeng, L., Jiajia, Z., & Guorong, L. (2016, May)		X		X																X		
Gao, K. Z., Suganthan, P. N., Pan, Q. K., Tasgetiren, M. F., & Sadollah, A. (2016)	X	X		X					X			X										
Baykasoğlu, A., & Karaslan, F. S. (2017)		X		X								X			X				X	X		
Salido, M. A., Escamilla, J., Barber, F., & Giret, A. (2017)		X		X																X		
Li, X., Peng, Z., Du, B., Guo, J., Xu, W., & Zhuang, K. (2017)		X		X								X								X		
Zhang, S., & Wong, T. N. (2017).	X	X		X								X			X					X		

Alagöz, O., & Azizoğlu, M. (2003).		X	X																	X		
Sabuncuoglu, I., & Kizilisik, O. B. (2003).	X	X		X					X											X		
Rangsaritratsamee, R., Ferrell Jr, W. G., & Kurz, M. B. (2004)		X		X								X										
Hall and Potts (2004)		X		X								X										
Adhitya, A., Srinivasan, R., & Karimi, I. A. (2004).		X		X	X											X						
Pfeiffer, A., Kádár, B., Csáji, B. C., & Monostori, L. (2005)		X		X								X										
Azizoglu, M., & Alagöz, O. (2005).		X	X																	X		
Wong, T. N., Leung, C. W., Mak, K. L., & Fung, R. Y. K. (2006)	X	X		X								X								X		
Lee, C. Y., Leung, J. Y., & Yu, G. (2006)		X		X																X		
Yang, B. (2007).		X		X								X										
Sawik, T. (2007)		X		X								X										
Hall, N. G., Liu, Z., & Potts, C. N. (2007)		X		X								X										
Yuan, J., & Mu, Y. (2007)		X		X																		X
Liu, L., Gu, H. Y., & Xi, Y. G. (2007)		X	X																	X		
Silva, C. A., Sousa, J. M. C., & Runkler, T. A. (2008)		X	X	X								X										
Arnaout, J. P., & Rabadi, G. (2008)		X		X																X		
Zuo, X., Mo, H., & Wu, J. (2009)	X			X					X													
Zhou, R., Nee, A. Y. C., & Lee, H. P. (2009).		X		X								X	X									
Naseri, E., & Kuzgunkaya, O. (2010)		X										X			X					X		
Hall, N. G., & Potts, C. N. (2010).		X		X																		
Corominas, A., & Pastor, R. (2010)		X				X																
Arnaout, J. P. (2010)		X																		X		
Gomes, M. C., Barbosa-Póvoa, A. P., & Novais, A. Q. (2010).		X		X								X										

Nouiri, M., Bekrar, A., Jemai, A., Ammari, A. C., & Niar, S. (2018)		X		X																X		
Valledor, P., Gomez, A., Priore, P., & Puente, J. (2018)		X		X					X			X								X		
Wang, D., Yin, Y., & Cheng, T. C. E. (2018)		X		X																X		
Ma, Z., Yang, Z., Liu, S., & Wu, S. (2018).		X						X	X	X										X		
Gao, K., Yang, F., Zhou, M., Pan, Q., & Suganthan, P. N. (2018)		X		X											X							
Buddala, R., & Mahapatra, S. S. (2019)		X		X																X		
Peng, K., Pan, Q. K., Gao, L., Li, X., Das, S., & Zhang, B. (2019)		X		X								X	X							X		
Jiang, Y., Zhou, X., & Chen, Y. (2019)		X																			X	
Larsen, R., & Pranzo, M. (2019)		X		X					X													

Tabla 1: Artículos de programación y reprogramación con incertidumbre.
Fuente: Elaboración propia

En su mayoría, estas investigaciones abordan el tema de programación y reprogramación con incertidumbres relacionadas con el averío de máquinas y la cancelación y llegada de pedidos. Por otro lado, en estas investigaciones la tendencia para los métodos en la solución de estos problemas es realizarlas en un entorno dinámico y no en uno estático, debido a que la producción en la realidad está envuelta en un ambiente dinámico y es inevitable que ocurran eventos inesperados en la ejecución de la producción.

A continuación, se presentará una clasificación taxonómica de diferentes publicaciones teniendo en cuenta el entorno en que se desarrolló el problema (Single Machine, Parallel Machine, Flow shop, Job shop, Flexible Flow Shop, Flexible Job Shop), los métodos de solución utilizados y los autores de cada uno.

3.4.1. Single Machine

Para el problema de programación y reprogramación de la producción con incertidumbre la mayoría de las investigaciones se desarrollaron en entornos de Single Machine y Job Shop. Las incertidumbres con mayor análisis en estas investigaciones fueron por avería de máquinas y llegada y/o cancelación de trabajos.

En las investigaciones de Single Machine se utilizaron como métodos de solución reglas de despacho y heurísticas, pero en su mayoría fueron utilizadas heurísticas como algoritmo genético, el problema del agente viajero, búsqueda local y otras propuestas por diferentes autores.

Authors	Solution Methods																			
	Rules						Heuristics													
	FIFO	SPT	EDD	ERD	Least tool changes	Fixed secuence	Algorithm GAA	A heuristic based on very large scale neighborhood (VLSN) search.	A fully polynomial time approximation scheme (FPTAS) algorithm	Stability and utility of a single event	Branch-and-bound Algorithm	PQR tree	β-Heuristic	Two-stage multi-population genetic algorithm	Apparent tardiness cost (ATC)	Optimized Surrogate Measure Heuristic	Travelling Salesman	Local search	Algorithm LIST	Algorithm CMHEU	Match-up Algorithm
Farn, C. K., & Muhlemann, A. P. (1979)	X				X												X				
Bean, J. C., Birge, J. R., Mittenthal, J., & Noon, C. E. (1991)																					X
Church & Uzsoy (1992)			X																		
BAPTISTE, P., & FAVREL, J. (1993).												X									
Wu, Storer &Pei-Chan (1993)						X												X			
Daniels, R. L., & Kouvelis, P. (1995)											X										
Unal, A. T., Uzsoy, R., & Kiran, A. S. (1997)																			X	X	
Daniels, R. L., & Carrillo, J. E. (1997)													X								
O'Donovan, R., Uzsoy, R., & McKay, K. N. (1999).															X	X					
Viera, Herrman & Lin (2000)	X																				
Cowling, P., & Johansson, M. (2002)										X											
Hall and Potts (2004)		X	X																		
Yang, B. (2007).								X													
Hall, N. G., Liu, Z., & Potts, C. N. (2007)											X										
Yuan, J., & Mu, Y. (2007)				X																	
Liu, L., Gu, H. Y., & Xi, Y. G. (2007)														X							
Hall, N. G., & Potts, C. N. (2010).							X														
Liu, Z., & Ro, Y. K. (2014)									X												

Tabla 2: Artículos de Single Machine.
Fuente: Elaboración propia

3.4.2. Parallel Machines

Para el caso de Parallel Machines al igual que Single Machine en estas publicaciones los métodos de solución utilizados fueron reglas de despacho y heurísticas. Cabe resaltar que la mayoría de estas publicaciones analizaron como perturbación del sistema la avería de máquinas.

Authors	Solution Methods											
	Rules				Heuristics							
	FIFO	SPT	EDD	Minimum Weighted Cmax Difference (MWCD)	Branch-and-price method	Pseudo-polynomial DP algorithm	ESGA Algorithm	Pseudo polynomial algorithm	Complete Rescheduling (MIP)	Machine Breakdown	Polynomial time heuristic H1	branch and bound procedures BAB1 and BAB2
Church & Uzsoy (1992)			X									
Vieira, G. E., Herrmann, J. W., & Lin, E. (2000).	X											
Alagöz, O., & Azizoğlu, M. (2003).											X	X
Azizoglu, M., & Alagöz, O. (2005).		X										
Lee, C. Y., Leung, J. Y., & Yu, G. (2006)								X				
Arnaout, J. P., & Rabadi, G. (2008)									X			
Arnaout, J. P. (2010)				X					X			
Gürel, S., & Cincioğlu, D. (2015)							X					
Wang, D., Yin, Y., & Cheng, T. C. E. (2018)					X	X						

Tabla 3: Artículos de Parallel Machine.
Fuente: Elaboración propia

3.4.3. Flow Shop

Para el problema de tipo Flow Shop a diferencia de Single Machine y Parallel Machine, no solo utiliza como métodos de solución reglas de despacho y heurísticas, si no que ya se incluyen como

métodos de solución metaheurísticas y un método de programación entera con relajación lagrangeana.

En estas investigaciones los problemas incluyen entre dos a cuatro incertidumbres entre la cuales se destaca averías de máquinas. De esta forma los métodos de solución son cada vez más complejos debido a la presencia de diferentes tipos de incertidumbres.

Authors	Solution Methods								
	Rules		Heuristics					Metaheuristics	Other Methods
	FIFO	Jhonson Rule	A discrete teaching-learning-based optimisation (DTLBO)	Predictive-reactive approach event-driven rescheduling policy	Match-up Algorithm	Genetic algorithm based solver	Genetic algorithm	Multi-objective rescheduling architecture	Integer Programming. Lagrangean Relaxation
Chang, S. C., & Hsieh, F. S. (1992)									X
Guo, B., & Nonaka, Y. (1999)		X							
Akturk, M. S., & Gorgulu, E. (1999)					X				
Dupon, A., Van Nieuwenhuyse, I., & Vandaele, N. (2002).	X								
Katragjini, K., Vallada, E., & Ruiz, R. (2013).				X					
Li, J. Q., Pan, Q. K., & Mao, K. (2015)			X						
Valledor, P., Gomez, A., Priore, P., & Puente, J. (2018)								X	
Ma, Z., Yang, Z., Liu, S., & Wu, S. (2018).						X			

Tabla 4: Artículos de Flow Shop.
Fuente: Elaboración propia

3.4.4. Job Shop

La mayoría de los artículos de Job Shop fueron abordados en la literatura mediante métodos de solución de reglas de despacho, como se muestra a continuación:

Authors	Solution Methods													
	Rules													
	Randow	FIFO	SPT	EDD	Earliest operational Due Date	Earliest latest start time	Least work remaining	Minimum critical ratio	Least operational slack	COM	Fluctuation Smoothing Policy for Variance of Cycle-Time	Least Slack	A dispatching rule MDT(Maximum Delivery Time)	Dispatching rules
Muhlemann, A. P., Lockett, A. G., & Farn, C. K. (1982).	X	X	X	X	X	X	X	X	X	X				
Christy, D. P., & Kanet, J. J. (1988)			X		X									
Kumar, P. R. (1994).											X	X		
Chang, J. W., & Luh, Y. P. (1997)													X	
Sabuncuoglu, I., & Bayız, M. (2000)			X											X

Tabla 5. Artículos de Job Shop con métodos de solución de reglas de despacho.
Fuente: Elaboración Propia

Los métodos de solución utilizados para resolver este tipo de problemas comprenden diferentes tipos de reglas de despacho, heurísticas y metaheurísticas, y en ocasiones se utilizan varios métodos para comparar los resultados o la combinación de estos para resolver el problema. Sin embargo, se destaca el uso de heurísticas como el método de solución más utilizado (ver tabla 6).

La mayoría de estos artículos contienen los dos tipos de incertidumbre más utilizados en la revisión de estas publicaciones, los cuales son la llegada y/o cancelación de pedidos y la avería de máquinas.

Authors	Solution Methods																							
	Heuristics																			Metaheuristics			
	Scheduling graph	A multi-objective variable neighborhood immune algorithm	Rescheduling expert simulation system	GRASP based approach	Non-monotonic Approach	Schedule stability Factor	A general simulation-based framework	PQR tree	Online Hybrid Agent-based Negotiation (oHAN)	Improved evolutionary algorithm (IEA)	AWI-J and AWI-O Algorithm	Reactive scheduling algorithm	Ant Colony Optimisation (ACO)	Variant of the generalized assignment problem	Beam search algorithm	Match-up Algorithm	Match up reactive scheduling approach	Optimized Surrogate Measure Heuristic	Linear Programming Based Heuristic	Genetic Algorithm	Affected Operations Rescheduling	scheduling binary-tree structure	Memetic Algorithm	Hybrid genetic algorithm and tabu search
BAPTISTE, P., & FAVREL, J. (1993).								X																
Li, R. K., Shyu, Y. T., & Adiga, S. (1993)																						X		
Jorge Leon, V., David Wu, S., & Storer, R. H. (1994)																				X				
Wu, H. H., & Li, R. K. (1995)	X																							
Abumaizar, R. J., & Svestka, J. A. (1997)																					X			
Byeon, E. S., Wu, S. D., & Storer, R. H. (1998)														X										
Mehta, S. V., & Uzsoy, R. M. (1998)																		X	X					
Bierwirth, C., & Mattfeld, D. C. (1999																				X				
Li, H., Li, Z., Li, L. X., & Hu, B. (2000)			X																					
Sabuncuoglu, I., & Bayiz, M. (2000)															X									
Sun, J., & Xue, D. (2001).																	X							
Bidot, J., Laborie, P., Beck, J. C., & Vidal, T. (2003)					X																			
Rangsaritratsamee, R., Ferrell Jr, W. G., & Kurz, M. B. (2004)																				X				
Pfeiffer, A., Kádár, B., Csáji, B. C., & Monostori, L. (2005)						X																		
Wong, T. N., Leung, C. W., Mak, K. L., & Fung, R. Y. K. (2006)									X															
Zuo, X., Mo, H., & Wu, J. (2009)		X																						
Zhou, R., Nee, A. Y. C., & Lee, H. P. (2009).													X											
Gomes, M. C., Barbosa-Póvoa, A. P., & Novais, A. Q. (2010).												X												
Dong, Y. H., & Jang, J. (2012).											X													
Zhang, L., Gao, L., & Li, X. (2013).																								X
Lv, S., & Qiao, L. (2014).										X														
Baykasoğlu, A., & Karaslan, F. S. (2017)				X																				
Salido, M. A., Escamilla, J., Barber, F., & Giret, A. (2017)																X							X	
Larsen, R., & Pranzo, M. (2019)							x																	

Tabla 6: Artículos de Job Shop con métodos de solución con heurísticas y metaheurísticas. Fuente: Elaboración propia

3.4.5. Flexible Job Shop

Dentro de las publicaciones de esta revisión literaria, el tipo de asignación de máquina Flexible Job Shop fue el tercero más utilizado. Ninguno de estos artículos tiene como enfoque la incertidumbre en lead time, los tipos de incertidumbre más utilizados en éstos son por averías de máquinas y llegada y/o cancelación de pedidos.

Los métodos de solución utilizados en estas publicaciones incluyen reglas de despacho, heurísticas y metaheurísticas, entre las cuales se destaca el uso de metaheurísticas

Authors	Métodos de solución														
	Rules			Heuristicas				Metaheuristicas							
	EDD	Fixed secuence	Dispatching rules	The multilevel-coding genetic algorithm	Partial rescheduling heuristic based on PSO	Active schedule generation	Algoritmo genetico	Djaya	Two-stage teaching-learning-based optimization	MODFJSSP	Hybrid Genetic Algorithm (HGA)	A hybrid multiagent negotiation/ant colony optimisation (HMA)	Two-stage artificial bee colony (TABC) algorithm	Hybrid artificial bee colony algorithm	Rolling horizon scheduling
YAMAMOTO & NOF (1985)		X	X			X									
Fang, J., & Xi, Y. (1997)	X														X
JAIN, A. K., & ELMARAGHY, H (1997)							X								
Al-Hinai, N., & ElMekkawy, T. Y. (2011).											X				
Shen, X. N., & Yao, X. (2015)										X					
Zhenyu, G., Linfeng, L., Jiajia, Z., & Guorong, L. (2016, May)				X											
Gao, K. Z., Suganthan, P. N., Pan, Q. K., Tasgetiren, M. F., & Sadollah, A. (2016)													X		
Li, X., Peng, Z., Du, B., Guo, J., Xu, W., & Zhuang, K. (2017)														X	
Zhang, S., & Wong, T. N. (2017).												X			
Nouiri, M., Bekrar, A., Jemai, A., Ammari, A. C., & Niar, S. (2018)					X										
Gao, K., Yang, F., Zhou, M., Pan, Q., & Suganthan, P. N. (2018)								X							
Buddala, R., & Mahapatra, S. S. (2019)									X						

Tabla 7: Artículos de Flexible Job Shop.
Fuente: Elaboración propia

3.4.6. Flexible Flow Shop

Este tipo de problema de asignación de máquina ha sido el menos utilizado en las publicaciones de esta revisión de la literatura. Los métodos de solución son heurísticas y metaheurísticas, algunas propuestas por los autores, otras basadas en unas ya existentes o por la combinación de dos o más heurísticas.

Authors	Métodos de solución			
	Heuristica		Metaheuristica	
	Algorithm REALL, RENON & REMAT	A variable neighborhood search (VNS) algorithm	A Multi-Start Variable Neighbourhood Descent	A decomposition-based approach with GA and SPT
Sawik, T. (2007)	X			
Choi, S. H., & Wang, K. (2012)				X
Rahmani, D., & Ramezanian, R. (2016).		X		
Peng, K., Pan, Q. K., Gao, L., Li, X., Das, S., & Zhang, B. (2019)			X	

Tabla 8: Artículos de Flexible Flow Shop.
Fuente: Elaboración propia

3.4.7. Análisis de la revisión de la literatura

En la revisión de literatura el problema de reprogramación de la producción fue abordado con diferentes tipos de incertidumbre. A continuación, se relaciona el análisis de los artículos con diferentes tipos de incertidumbre de esta revisión literaria:

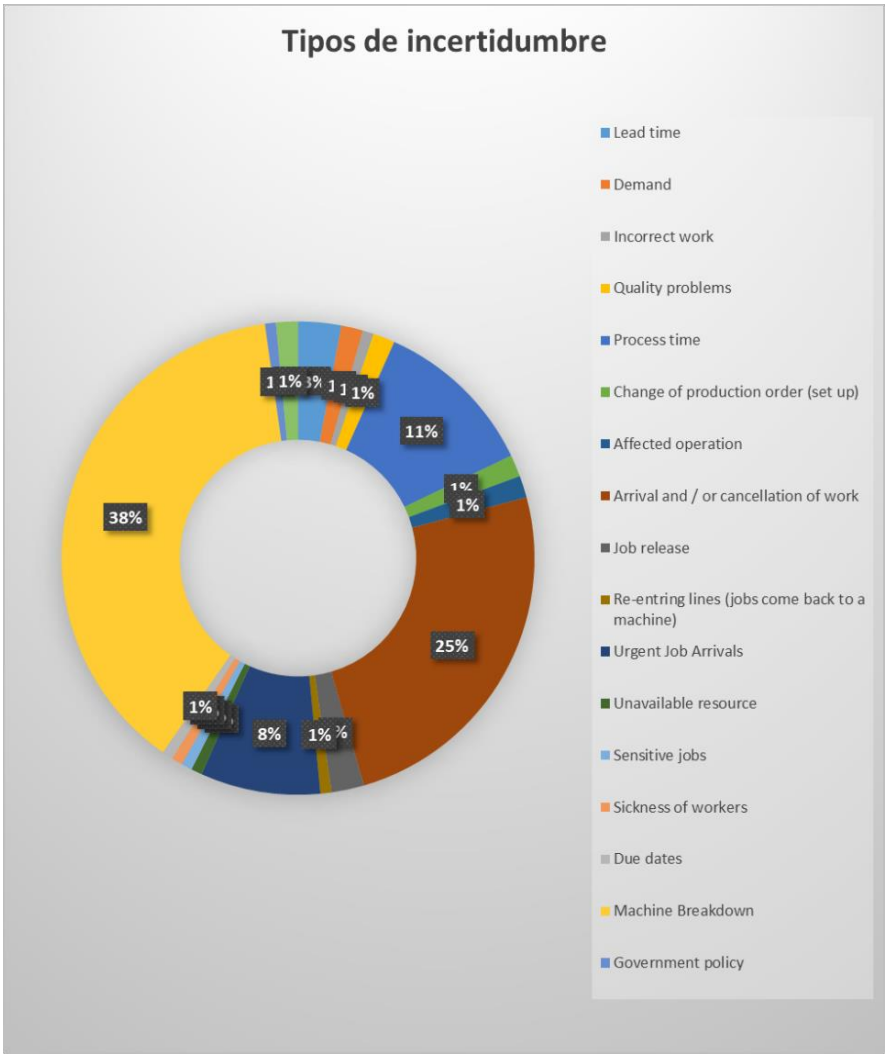

Figura 4: Tipos de incertidumbre.

Fuente: Elaboración propia.

De acuerdo con el análisis estadístico (Figura 4) los tipos de incertidumbre con mayor presencia en la literatura son: Averías de máquinas con un 38%, llegada y/o cancelación de pedidos con un 24% y tiempo de proceso con un 11%. Para el caso de estudio de esta investigación el lead time solo corresponde a un 3% de estas investigaciones.

Las investigaciones realizadas por Dupon et al (2002) y Dolgui et al (2002) enfocan el lead time como el intervalo de tiempo entre la hora de llegada de un trabajo al sistema y su hora de salida. Ninguna de las investigaciones de esta revisión tiene como foco la incertidumbre en la llegada de

materia prima y como afecta el plan de producción, por lo cual, es evidente la importancia de realizar investigaciones teniendo en cuenta esta incertidumbre y el desempeño logístico de Colombia en cuanto a la consecución de materias primas.

El entorno en el que se investigan y se proponen métodos de solución para resolver este problema de reprogramación de la producción se puede observar en el siguiente gráfico:

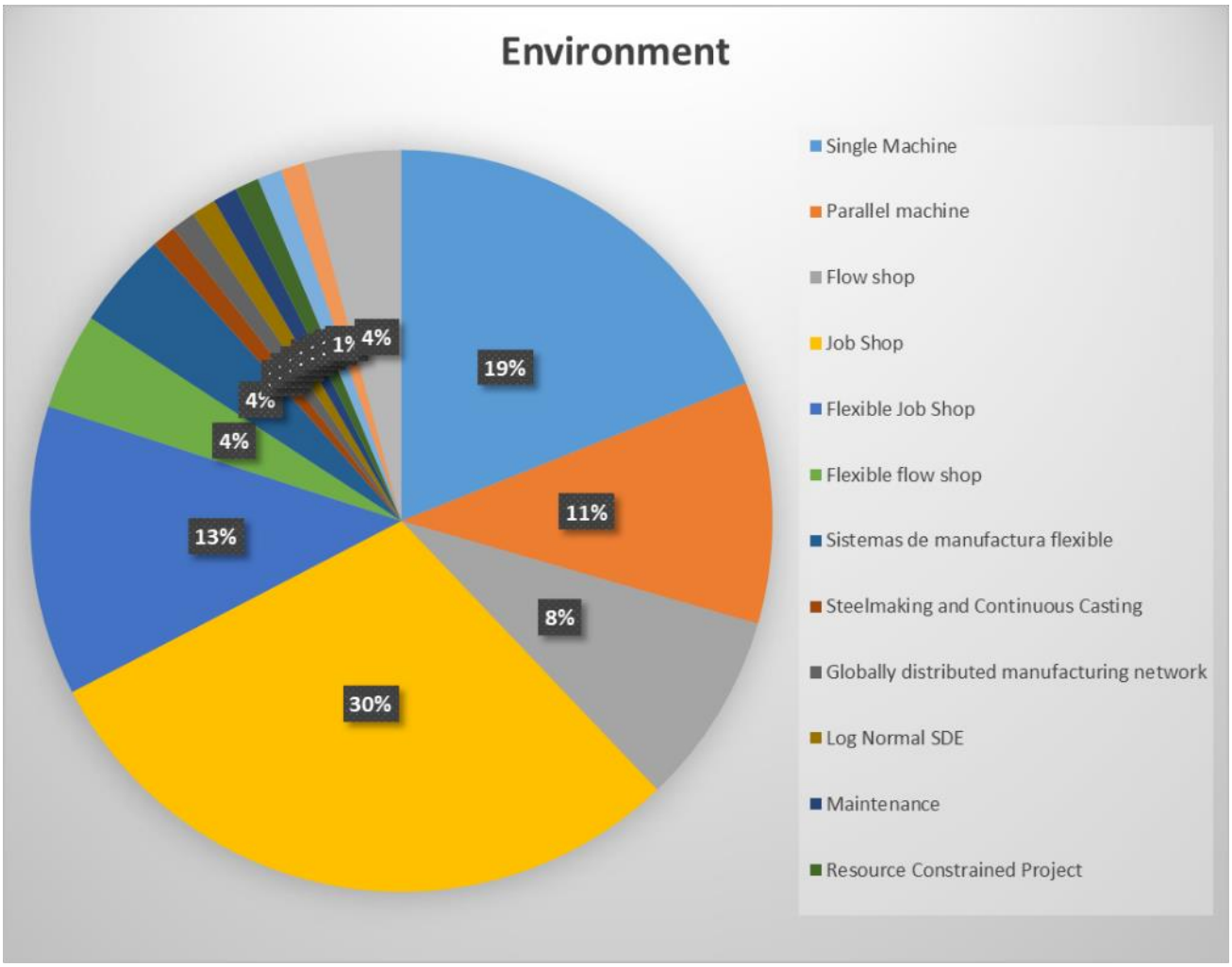

Figura 5: Tipos de problema de asignación de máquina y otros entornos.

Fuente: Elaboración propia

Los tipos de problema de asignación de máquina más utilizados según el análisis estadístico de estas publicaciones sobre programación y reprogramación con incertidumbre fueron: Job Shop con 29%, Single Machine con 19% y Flexible Job Shop con 13%.

Entre los tipos de problemas de asignación de máquina, las investigaciones de esta revisión literaria sobre reprogramación de la producción en ambientes de tipo Flexible Flow Shop (FFS) lograron un porcentaje de solo 4%. Debido a que ninguna de estas investigaciones con FFS considerar como incertidumbre el lead time en el aprovisionamiento de materias primas, construir un modelo basado en FFS para reprogramación de la producción teniendo en cuenta esta incertidumbre, constituye un aporte para reducir la brecha en las investigaciones de este tipo.

Los métodos de solución utilizados tanto para modelos de reprogramación con incertidumbre en lead time como los que son de tipo FFS incluyen reglas de despacho, heurísticas y metaheurísticas. A continuación, se relacionan los métodos de solución utilizados en las publicaciones de esta revisión literaria para incertidumbre con lead time:

- First in coming first in out (FIFO). (Dupon., Van Nieuwenhuyse, & Vandaele, 2002)
- Dynamic single-item and Multi-items optimization model (use of an auxiliary Markov chain). (Dolgui & Ould-Louly, 2002)
- Multi-Step Approach. (Adhitya, Srinivasan, & Karimi, 2004)
- Ecuación diferencial estocástica. (Shirai & Amano, 2014)

Como se ha evidenciado en el desarrollo de esta investigación existen muchos elementos aleatorios que pueden afectar los sistemas de producción y que no son fáciles de anticipar. Por lo cual, se hace necesario estrategias de mitigación de sus efectos, como lo son los esquemas de reprogramación. Con la reprogramación se obtiene una nueva solución al sistema y se debería asegurar que esta no sea completamente diferente a la inicialmente

planteada, debido a que la configuración del sistema responde a una planeación a mediano plazo que debe, en lo posible, respetarse.

De acuerdo con las investigaciones referentes al problema de reprogramación de máquina de tipo Flexible Flow Shop expuestas anteriormente, en el estado del arte, se pueden clasificar como se observa en la tabla 9, con el fin de hacer un análisis de como se ha abordado esta problemática.

De estas investigaciones cabe resaltar nuevamente que solo corresponden al 4% de la literatura encontrada sobre reprogramación de la producción, evidenciando que existe una brecha investigativa para ambientes de tipo Flexible Flow Shop (FFS).

Autor(es) – Año	Incertidumbre	Función objetivo	Algoritmo(s)	Descripción del/los Algoritmo(s)	Observaciones
Tadeusz Sawik (2007)	Modificar orden, cancelar orden y agregar orden	Minimizar el número de pedidos retrasados y el inventario de entrada y salida	REALL	Permite la reprogramación de todos los pedidos pendientes luego de la ruptura	Estos 3 algoritmos están basados en programación entera mixta y su elección depende del impacto que causen las interrupciones.
			RENNON	No reprograma los pedidos pendientes luego de la ruptura	
			REMAT	Para reprogramar solo un subconjunto de los pedidos de clientes restantes que están a la espera de suministros de materiales	
S. H. Choi, & K. Wang (2012)	Tiempos de procesamiento estocásticos	Minimizar Makespan	Un nuevo enfoque basado en la descomposición (DBA)	Este enfoque incluye un algoritmo propuesto K-means para convertir las estaciones en clústeres. Cada clúster se programa con SPT o con algoritmo genético dependiendo del cálculo de la diferencia del Makespan entre ambos.	

Donya Rahmani y Reza Ramezanian (2016)	Llegadas de nuevos trabajos	Minimizar la tardanza ponderada total	Búsqueda de vecindario variable	Además del objetivo inicial tiene en cuenta dos medidas de estabilidad y resistencia al cambio. La función por minimizar es Swt Este algoritmo tiene dos funciones de sacudida (shaking), que renueva la solución con su desplazamiento a otro vecindario local; y una función de búsqueda local que realiza una búsqueda precisa sobre la solución actual.	Debido a la complejidad computacional de SWT, se propone resolver este problema con el algoritmo de búsqueda de vecindario variable.
Kunkun Peng, Quan-Ke Pan, Liang Gao, Xinyu Li, Swagatam Das, Biao Zhang (2019)	Averías de máquinas, llegada de nuevos trabajos o variación en los tiempos de liberación de los trabajos	Minimizar el Makespan y la inestabilidad del sistema	Algoritmo de descenso de vecindario variable de arranque múltiple (MSVND)	El algoritmo MSVND está basado en el básico VND, en el cual se desarrolla una decodificación híbrida y se utilizan estrategias de intensificación y diversificación.	

Tabla 9. Investigaciones de reprogramación de la producción para FFS

Fuente: Elaboración propia

Estas investigaciones entre sí aunque tienen en común que abordan la problemática de reprogramación de la produccion en un ambiente FFS, enfocan este problema con diferentes funciones objetivo, incertidumbres y algoritmos particulares para solucionar la problemática de produccion asociada. Sepuede destacar que ninguna de estas investigaciones considera el lead time como el tipo de incertidumbre que afecta su plan de producción y en lo que se refiere a su función objetivo, aunque incluyen minimización de makespan o tardanza, ninguno considera una función bi-objetivo que permita la minimización de estas dos métricas

Por lo tanto, debido a las interrupciones que se pueden presentar en el sistema de producción, es necesario que en la nueva programación del sistema, las funciones objetivos a trabajar en el

problema de optimización conexo, deben apuntarle a la minimización de la diferencia entre la solución inicial y la reprogramada con el fin de no variar la parametrización de otros componenentes de soporte a la logistica de producción desarrollada de acuerdo a la programación inicial, conservando buenos niveles de desempeño en cuanto a las métricas del problema. Por lo cual, en esta investigación tiene por objetivo diseñar un modelo de reprogramación de la producción utilizando el tipo de problema de asignación de máquina de Flexible Flow Shop bajo condiciones de incertidumbre en el proceso de aprovisionamiento de materias primas que permita mejorar la confiabilidad sobre las métricas de Makespan y tardanza del programa de producción asociado.

4. OBJETIVOS

4.1. Objetivo general

Diseñar un modelo de reprogramación de la producción utilizando el tipo de problema de asignación de máquina de Flexible Flow Shop bajo condiciones de incertidumbre en el proceso de aprovisionamiento de materias primas que permita mejorar la confiabilidad sobre las métricas de Makespan y tardanza del programa de producción asociado.

4.2. Objetivos específicos

- Identificar los elementos con incertidumbre que pueden afectar un plan de producción y cuantificar los más significativos.
- Construir el estado del arte sobre modelos de programación y reprogramación existentes en condiciones de incertidumbre.
- Identificar las brechas de investigación en modelos de reprogramación de la producción en condiciones de incertidumbre.
- Diseñar un modelo de reprogramación de la producción con el uso de metaheurísticas para la minimización de la diferencia entre las métricas iniciales y las encontradas luego de la inclusión de la perturbación.

– Evaluar el desempeño del modelo en cuanto a tiempo de ejecución y soluciones halladas por otros métodos utilizados en este tipo de problema.

5. METODOLOGÍA DE INVESTIGACIÓN

El desarrollo de esta investigación se realizará en 3 fases:

- **Fase 1. Caracterización de los tipos de incertidumbre y entornos.** La investigación iniciará con la revisión del estado del arte de la programación y reprogramación de la producción, donde se identificará la incertidumbre y restricciones que presentan estos procesos.
- **Fase 2. Diseño de un modelo de reprogramación de la producción.** El modelo de reprogramación se planteará en términos de la minimización de la diferencia entre las métricas iniciales y las encontradas luego de la inclusión de la perturbación. El diseño de la estrategia de reprogramación se abordará utilizando métodos exactos y metaheurísticas. El modelo estará partiendo de una solución inicial del problema conocida.
- **Fase 3. Evaluación del modelo.** El diseño del modelo y el método de solución propuesto se evaluará con otros métodos utilizados en este tipo de problema. Para esta validación el modelo se comparará con las mejores soluciones de instancias conocidas. Esto permitirá conocer la habilidad del modelo en adaptarse a cambios que se presentan en las condiciones del entorno, y obtener mejores soluciones.

CAPITULO II: Modelación del problema de programación y reprogramación de la producción.

En esta sección se realiza una introducción a los esquemas de solución del problema de programación de Flow Shop. Se describe la modelación matemática de problemas tipo Flexible Flow Shop y sus esquemas de solución y luego, se aborda la temática referente a reprogramación de la producción en condiciones de incertidumbre donde se presenta una descripción de las investigaciones relacionadas a reprogramación de la producción para Flexible Flow Shop con el tipo de incertidumbre que consideraron, sus objetivos y el método de solución utilizado. Seguidamente, se explica el funcionamiento de los algoritmos desarrollados para programación y reprogramación de la producción para Flexible Flow Shop basados en el algoritmo genético de Chu-Beasley y una comparación de sus resultados con otros modelos similares encontrados en la literatura.

6. PROGRAMACIÓN DE LA PRODUCCIÓN EN AMBIENTE DE TIPO FLOW SHOP

El proceso de programación de la producción tiene como objetivo definir las secuencias de los trabajos, y a su vez asignar estos trabajos a las máquinas según su orden de procesamiento. Esta programación se debe hacer independientemente del ambiente, por ejemplo, Flow Shops, con el objetivo de conocer la secuenciación de tareas óptima basado en una, o varias, métricas de desempeño. En este caso, la programación se caracteriza porque los trabajos tienen el mismo orden de procesamiento y los tiempos de ejecución varían generalmente en cada una de las etapas (Morales, 2012). Debido a su complejidad, este problema está clasificado como **NP-Hard**, pero se pueden encontrar soluciones exactas para cálculo de Makespan con tamaños pequeños de trabajos y máquinas (*n y m*) (Taillard E. , 1990).

En general, los ambientes Flow Shop se describen como un problema en el cual *n*-trabajos o tareas deben ser procesadas por un conjunto de *m*-máquinas distintas, estas tareas deben tener el mismo flujo de procesamiento o secuencia tecnológica en las máquinas. (Ocampo, Grisales, & Echeverri, 2006). Debido a que las tareas tienen el mismo orden o flujo de procesamiento este

tipo de problema es combinatorio por las diferentes posibles secuencias según la cantidad *n* de trabajos, por lo cual el número total de secuencias es *n!.* Dentro de la literatura, también se puede encontrar este problema como Flow-Shop no permutado cuando se permiten cambios en el orden del trabajo en diferentes máquinas. (Xiao, Yuan, Konak, & Zhang, 2015). Para el desarrollo de este capítulo solo se referirá a Flow Shop permutado. El problema clásico de Flow Shop tiene por objetivo minimizar el Makespan de los *n*-trabajos a través de las *m* máquinas. Otras medidas de desempeño o métricas, a parte del Makespan, para este tipo de problema son (Morales, 2012):

- Tiempo de procesamiento (Tiempo de flujo)
- Tiempo de flujo total
- Número de trabajo de tardíos
- Tardanza total
- Tardanza máxima

6.1. El problema de la secuenciación de Flow Shop

En este problema de secuenciación se consideran *n*-trabajos que se procesan en el mismo orden en *m*-máquinas (figura 6), el tiempo de procesamiento del trabajo *i* en la máquina *j* viene dado por p_{ij}, donde *i*=1..., n; y *j*=1…, m. Este tiempo p_{ij} es fijo y es no negativo, en algunos casos es cero (0) si el trabajo no necesita ser procesado en alguna máquina. Teniendo en cuenta que el objetivo de este problema es minimizar el Makespan, se considera lo siguiente (Taillard E. , 1990):

- Cada trabajo debe ser procesado por una máquina cada vez.
- Cada trabajo debe ser procesado en el orden máquina 1, 2, . . ., m.
- Cada máquina procesa un solo trabajo a la vez.
- Los trabajos tienen la misma opción de ser programados. No hay operaciones prioritarias.
- Los tiempos de procesamiento de los trabajos en las diferentes máquinas son determinados y fijos.

– Los tiempos de preparación de las operaciones en las distintas máquinas están incluidos en los tiempos de procesamiento.

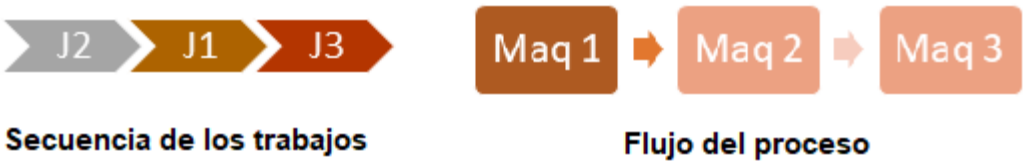

Figura 6. Flow Shop.

Fuente Elaboración propia.

Como resultado de resolver este problema obtendremos una secuencia y el Makespan mínimo que puede ser óptimo en el caso de que se tengan pocas máquinas y trabajos (Conway, Miller, & Maxwell, 2003), o puede ser una posible solución obtenidas con el uso de heurísticas y metaheurísticas cuando el problema sea más grande. En la figura 7 se puede observar una solución típica de un problema de programación en ambientes tipo Flow Shop resuelto con la heurística NEH para 5 trabajos y 4 máquinas.

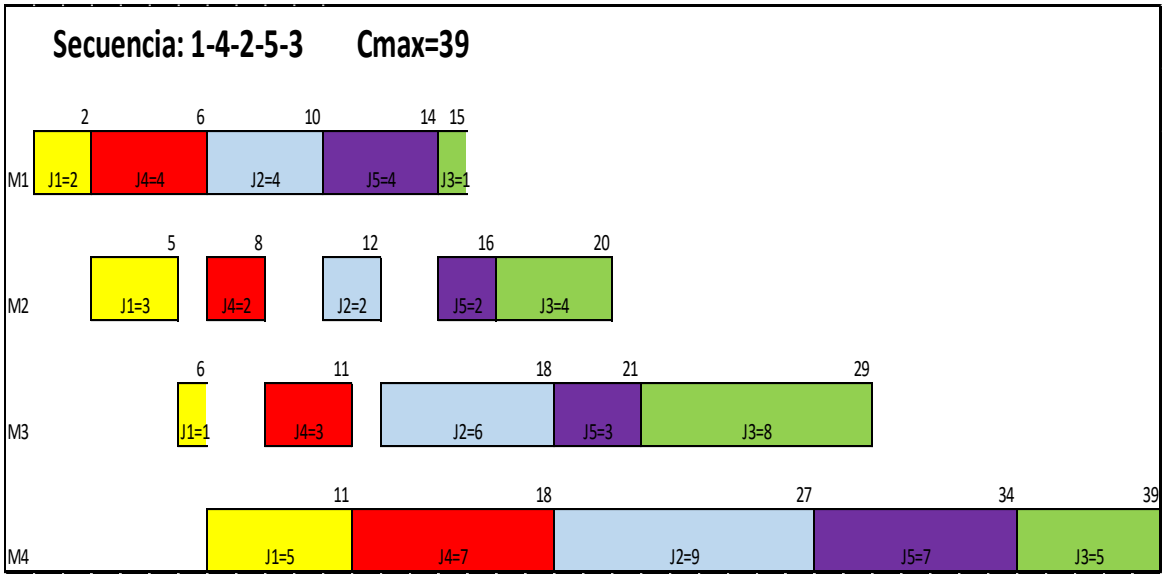

Figura 7: Ejemplo de un problema de tipo Flow Shop resuelto con algoritmo NEH

Fuente: Elaboración propia

6.2. Modelación matemática del problema de programación para sistemas tipo Flow shop

Para calcular el Makespan, a partir de los tiempos de procesamiento de los trabajos en cada máquina (p_{ij}) y la secuencia de los trabajos $\{J_1, J_2, \dots J_n\ \}$, se calcula el inicio de cada trabajo en la máquina 1, partiendo del trabajo 1, con la ecuación (1) (Ocampo, Grisales, & Echeverri, 2006):

$$C(J_1, 1) = p(J_1, 1) \qquad (1)$$

Se calculan los tiempos de ejecución de los otros trabajos *i* diferentes a J_1 en la máquina 1 con la ecuación (2), respetando la secuencia de los trabajos.

$$C(J_i, 1)\ =\ C(J_{i-1}, 1)\ +\ p(J_i\ ,1) \qquad desde\ i\ =\ 2, \dots, n \qquad (2)$$

Se calcula el tiempo de ejecución del trabajo 1 en el resto de las máquinas teniendo en cuenta que el tiempo de ejecución de este trabajo en cada máquina *j* se le debe sumar el tiempo que tardo en ejecutarse este mismo trabajo en la máquina anterior (*j-1*), se calcula con la siguiente ecuación (3)

$$C(\ J_1, j\ ) = C(J_1, j-1) +\ p\ (J_1, j)\ desde\ j = 2, \dots, m \qquad (3)$$

Con las ecuaciones anteriores se obtiene el tiempo de ejecución del trabajo 1 en todas las máquinas y el tiempo de ejecución del resto de trabajos en la máquina 1. A partir de estos datos se puede calcular el tiempo acumulado de la secuencia de producción hasta que finaliza el último trabajo en la última máquina. Para calcular este tiempo acumulado se utiliza la ecuación (4) y se debe considerar que para calcular este tiempo en un trabajo *i* se debe elegir el máximo entre el tiempo acumulado del trabajo anterior J_{i-1} en la máquina j y el tiempo acumulado del trabajo *i* en la máquina anterior (*j-1)*.

$$C(\ J_i, j\ ) = max\{C(J_{i-1}, j), C(J_i, j-1)\} +\ p\ (J_1, j) \text{ para i} = 2, \dots, \text{n y j} = 2, \dots, \text{m} \qquad (4)$$

Para finalizar con la ecuación (5) se obtiene el Makespan de la secuencia desde que ingresa el trabajo J_1 hasta que sale el trabajo J_n de la máquina m.

$$Cmax = C(J_{n,m}) \quad (5)$$

6.3. Esquemas de solución generales para el problema de programación para sistemas tipo Flow shop.

A continuación, se presentan alternativas de solución al problema de programación para ambientes Flow Shop.

- **Algoritmo NEH**

Este algoritmo fue desarrollado por Nawaz, Enscore y Ham en 1983, está basado sobre la premisa de que un trabajo con mayor tiempo total de proceso en todas las máquinas debe tener mayor prioridad que un trabajo con menor tiempo total de proceso. El procedimiento en general tiene los siguientes pasos (Nawaz, Enscore Jr, & Ham, 1983):

- Se ordenan los trabajos de mayor a menor teniendo en cuenta la suma de los tiempos de procesamiento en las máquinas. El tiempo de procesamiento para cada trabajo se calcula con:

$$T_{i=} \sum_{j=1}^{m} t_{i,j} \quad (6)$$

- Se escogen los dos primeros trabajos de la lista y se organizan calculando el Makespan de las dos posibles secuencias de forma que se escoja la secuencia donde el Makespan sea el mínimo. No se puede cambiar el orden relativo de estos dos trabajos en el resto del algoritmo.
- Se inserta el trabajo que sigue en la lista en la secuencia parcial encontrada en paso 2 en todas las posiciones posibles buscando que se minimice el Makespan.
- Cuando $i = n$ (n es el número total de trabajos) se detiene el algoritmo de lo contrario, $i = i + 1$ y se realiza el paso 3.

- **Tabu Search**

Es una metaheurística de optimización global propuesta por Grover que consiste en explorar el espacio de búsqueda partiendo de una solución a su mejor vecino pese a que este resultado pueda generar un deterioro en el valor de la función objetivo. Esto evita que las soluciones de esta metaheurísticas queden atrapadas en un óptimo local, de esta forma se generan nuevos vecinos o soluciones y se examinan los valores de la función objetivo (Barbarosoglu & Ozgur, 1999).

Para evitar en lo posible el ciclo, se asocia un elemento *t* a un movimiento realizado en la iteración *i*, con este elemento se define un conjunto de movimientos que ahora son tabú (prohibidos) y se almacenan en un conjunto *T* llamado lista tabú. La lista tabú es una cola ordenada de estos movimientos prohibidos y cada vez que se realiza un movimiento, se inserta al final de la lista tabú y se elimina el primer elemento de la lista. En su forma más simple, la búsqueda tabú requiere lo siguiente (Ben-Daya & Al-Fawzan, 1998):

- Secuencia inicial
- Mecanismo para generar alguna vecindad de la secuencia actual
- Lista tabú
- Criterios de parada.

Algunos criterios adicionales son: 1) Criterio de aspiración, si el movimiento tabú cumple algún criterio como un mejor valor de la función objetivo; 2) Esquema de intensificación, donde se enfoca más insistentemente en las regiones buenas y 3) Esquema de diversificación, la búsqueda se realiza en nuevas áreas que no se habían explorado antes.

El procedimiento genérico de Tabu Search es el siguiente (Taillard E. , 1990):

- Comienza con alguna solución factible x_0, y la lista tabú T comienza vacía (0).

$$x^* = x_0, esta\ es\ la\ mejor\ solución\ encontrada\ hasta\ ahora$$
$$C^* = C(x_0), es\ el\ valor\ objetivo\ de\ la\ solución\ x^*$$
$$k = 0$$

- En $M(x_k)$ se elige m, un movimiento transformando x_k y que minimice $C\big(m(x_k)\big)$. Este movimiento no puede ser prohibido, es decir que no debe estar en la lista tabú. Este movimiento se elige de todos los elementos posibles o parciales de $M(x_k)$. Entonces, $x_{k+1} = m(x_k)$.
- Si $C(x_{k+1}) < C^*$, entonces $C^* = C(x_{k+1})$ y $x^* = x_{k+1}$.
- S es la cantidad permitida de número de movimientos en la lista tabú T. Si $T = S$, se remueve el movimiento más viejo y se adiciona el movimiento t definido por m y x_{k+1}.
- Si no se cumple el criterio de parada se devuelve al punto 2.

- **Algoritmo genético de Chu-Beasley (AGCB)**

El AGCB es una versión modificada del algoritmo genético básico. Su principal característica consiste en mantener constante el tamaño de la población de alternativas de solución En cada iteración solo se reemplaza un individuo de la población a la vez. Esta estrategia permite encontrar múltiples soluciones y conservar la diversidad del conjunto de alternativas de manera que se evite caer en soluciones locales sub-óptimas (Ocampo, Grisales, & Echeverri, 2006).

Los pasos para este algoritmo se describen a continuación (Chu & Beasley, 1997):

Paso 1. Se genera una población inicial de manera aleatoria.
Paso 2. Se evalúa el fitness de la solución.
Paso 3. Se eligen dos soluciones padres para la reproducción. La mejor solución o el individuo más apto se elige por torneo binario. Para producir un hijo se deben realizar dos torneos binarios.
Paso 4. Se genera una nueva solución hijo usando un operador de cruce entre los dos padres. Se genera un punto aleatorio de cruce, por lo que la solución hijo consistirá en los primeros genes p tomados del primer padre y los genes restantes $(n - p)$ tomados del segundo padre o viceversa. Para que se garantice la legitimidad de este hijo en un problema tipo Flow Shop se

puede aplicar una variación a este operador, llamada recombinación PMX (Partially Mapped Crossover), que consiste en lo siguiente (Ocampo, Grisales, & Echeverri, 2006):

a. Se elige de forma aleatoria dos puntos de cruce.
b. Estos 2 segmentos se intercambian en los dos hijos que se generan.
c. Para cada hijo el resto de las cadenas se obtienen haciendo búsqueda o mapeo entre los 2 padres. De manera que, si un valor no está contenido en el segmento intercambiado, permanece igual, de lo contrario, se sustituye por el valor que tenga dicho segmento en el otro padre.

Paso 5. Se realiza un proceso de mutación donde hay un intercambio de elementos en dos genes seleccionados al azar. En el caso de Flow Shop esta mutación consiste en escoger, de manera aleatoria, dos trabajos e intercambiarlos en la secuencia de producción. Si la solución obtenida de esta mutación tiene un valor fitness mejor que el hijo obtenido del paso 4, se escoge esta nueva solución. De lo contrario se mantiene la solución obtenida en el paso 4.

6.4. Desarrollo de algoritmos genéticos para el problema de programación de máquinas de tipo Flow Shop

Uno de los esquemas de solución para el problema que ha venido tomando mucha relevancia es mediante la construcción de Algoritmos Genéticos. En este caso se presenta una aproximación del problema basado en el propuesto por Chu y Beasley para resolver el problema de programación de máquinas de tipo Flow Shop con el objetivo de encontrar la mejor secuencia con el menor Makespan. Además, se usa Tabu Search con el objetivo de obtener una mejor secuencia que de un mejor valor fitness a partir de la secuencia elegida de la población inicial y de esta forma obtener los padres. Los datos para evaluar en este algoritmo fueron tomados de las instancias conocidas de Taillard (ta001-005) para un problema de 20 trabajos con 5 máquinas.

El esquema de este algoritmo consiste en:

Paso 1. Generar una población con un numero P de las posibles soluciones de $n!$.

Paso 2. Se calcula el Makespan, que en este caso es el fitness del algoritmo, de todas las secuencias P. El cálculo del Makespan se realiza según el procedimiento expuesto anteriormente

en el modelo matemático, donde a partir del tiempo del primer trabajo de la secuencia en la máquina #1 se pueden calcular los tiempos de ejecución del resto de trabajos en la máquina #1 y el tiempo de ejecución acumulado del primer trabajo de la secuencia en el resto de máquinas; posteriormente, se puede calcular el tiempo de ejecución acumulado del resto de trabajos en las otras máquinas, y de esta manera obtener el Makespan del total de la secuencia (Ocampo, Grisales, & Echeverri, 2006).

Se elige la secuencia p de P con el mejor valor fitness, si hay dos o más secuencias con este mismo valor fitness, se elige una aleatoriamente. De esta forma obtenemos el padre inicial #1 que posteriormente se le aplicará Tabu Search para obtener el padre #1.

Nota: Para el resto del algoritmo donde se calcule el fitness y como resultado obtengamos dos o más secuencias con el mismo valor de fitness, se elige una de éstas de manera aleatoria.

Paso 3. Se realiza nuevamente los pasos (1) y (2) para obtener el padre inicial #2.

Paso 4. Se aplica Tabu Search para los dos padres iniciales con el objetivo de obtener los dos padres (Padre #1 y Padre #2) a los que se le realizará cruzamiento de esta forma:

a) A partir del padre inicial el algoritmo de Tabu Search busca 3 movimientos tabú que son los que dan mejores resultados de Makespan para cada 20 posibles movimientos, y así obtener una lista tabú con 3 movimientos.
b) Posteriormente se realiza un ciclo "*for*" de 50 iteraciones usando la última secuencia y la lista tabú con 3 movimientos obtenidas del paso (a). Se continúa en la búsqueda del mejor resultado para Makespan de cada 20 movimientos posibles y se actualiza la lista tabú.
c) La secuencia y la lista tabú para el inicio de cada iteración diferente a la #1 provienen de la iteración anterior. El algoritmo finaliza cuando alcanza las 50 iteraciones.
d) Tabu Search se aplica de manera individual para cada padre inicial, dando como resultado un padre para cada uno, el Padre #1 y posteriormente el Padre #2.

Nota: De este paso hasta el 8 se introduce en un ciclo "*for*" con el objetivo de generar un numero *i* de iteraciones, dando como resultado *i* posibles secuencias para resolver este problema de programación de máquinas en ambiente Flow Shop.

Paso 5. Se realiza el cruzamiento de Padre #1 y Padre # 2 esto da como resultado dos hijos y se escoge el de mejor fitness. El cruzamiento en el algoritmo se realiza de la siguiente forma:

a) Del Padre #1 se toman dos bloques las tareas ubicadas dentro de la secuencia en las posiciones 5 hasta 8 y en las posiciones 13 hasta 16 y se insertan estos dos bloques en el Padre #2 en las mismas posiciones. Como se muestra en la figura 8:

Posición	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	19
Padre #1	6	4	11	16	15	18	1	17	33	9	14	8	2	19	5	7	12	10	13	20
Padre #2	4	9	5	14	19	1	7	1	6	8	16	13	15	11	10	12	2	3	18	20

Cruce Padre					Padre #2								Padre #2							
#1	6	4	11	16	19	1	7	1	33	9	14	8	15	11	10	12	12	10	13	20

Cruce Padre					Padre #1								Padre #1							
#2	4	9	5	14	15	18	1	17	6	8	16	13	2	19	5	7	2	3	18	20

Figura 8. Cruzamiento de Padre #1 con el Padre #2

Fuente: Elaboración propia

b) Luego se identifican los números repetidos de los trabajos que quedan del padre original con los dos bloques nuevos insertados. Estos trabajos repetidos del padre original se “eliminan” y haciendo una comparación entre los dos bloques del padre original y los dos bloques nuevos insertados se identifican aquellos que no se repiten. Luego se procede a colocar los trabajos de los bloques del padre original que no se repiten dentro de la secuencia hijo en las posiciones que se habían “eliminado” anteriormente (*), respetando el orden en el que estaban originalmente.

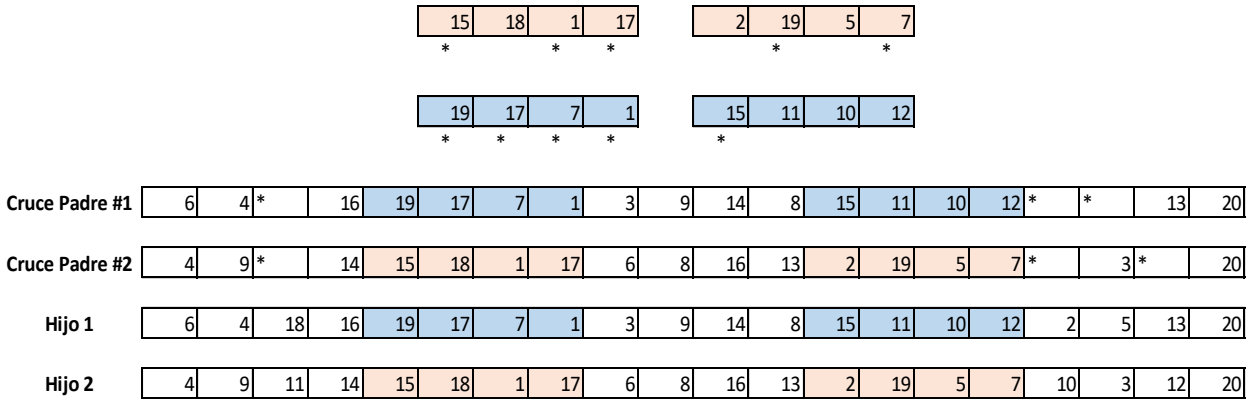

Figura 9. Hijos #1 y #2 originados por el cruzamiento de dos padres
Fuente: Elaboración propia

c) Como resultados obtenemos 2 soluciones hijos.

Paso 6. Se calcula el Makespan de las dos soluciones hijos y se escoge el de menor resultado.

Paso 7. Se realiza mutación según una probabilidad determinada al hijo escogido en el paso anterior al intercambiar dos trabajos dentro de esta secuencia. Si al calcular el Makespan de la secuencia de la mutación es mejor que el resultado del hijo se conserva como solución la secuencia de la mutación, de lo contrario, permanece como solución la secuencia hijo.

Paso 8. Se repite desde el paso 4 al 7 hasta que se cumpla el número total de iteraciones establecidas en el algoritmo. Los padres iniciales #1 y #2 son los mismos para cada iteración

Paso 9. Para finalizar, se calcula el Makespan de todas las soluciones y se identifica la secuencia con el menor Makespan. Si hay dos o más secuencias con este mismo valor de Makespan, se escoge al azar una de las secuencias.

En la sección de Anexos se presenta el algoritmo genético para Flow Shop con el objetivo de minimizar el Makespan desarrollado con el programa **MATLAB.**

6.5. Análisis de resultados del modelo de programación

A continuación, se muestran los resultados del algoritmo con los siguientes datos obtenidos de las instancias de Taillard (ta001-005) para 20 trabajos y 5 máquinas:

J/M	1	2	3	4	5
1	54	79	16	66	58
2	83	3	89	58	56
3	15	11	49	31	20
4	71	99	15	68	85
5	77	56	89	78	53
6	36	70	45	91	35
7	53	99	60	13	53
8	38	60	23	59	41
9	27	5	57	49	69
10	87	56	64	85	13
11	76	3	7	85	86
12	91	61	1	9	72
13	14	73	63	39	8
14	29	75	41	41	49
15	12	47	63	56	47
16	77	14	47	40	87
17	32	21	26	54	58
18	87	86	75	77	18
19	68	5	77	51	68
20	94	77	40	31	28

Tabla 10. Tiempos de procesamiento para 20 trabajos y 5 máquinas
Fuente: Instancias de Taillard. (Taillard E. , 1993)

Se mantuvo el tamaño de la población inicial $P = 500$ y el número de iteraciones en Tabu Search, $W = 50$ constante para todas las corridas. La probabilidad de mutación se mantuvo en 0.4 en todas las corridas. Dentro del algoritmo genético se varía el número de iteraciones i donde está incluido Tabu Search, cruzamiento y mutación. A continuación, se presentan los resultados.

i	*Tiempo*	Cmax	Secuencia
1000	8 horas	1306	8-4-3-11-17-9-6-16-13-5-14-1-19-12-15-18-2-10-7-20
2000	4,7 horas	1302	3-15-17-14-1-11-8-9-6-13-7-16-18-5-4-19-2-10-12-20
5000	7,8 horas	1297	8-9-15-5-19-6-17-7-16-18-4-11-14-10-12-3-1-2-13-20
10000	13,25 horas	1297	8-17-16-6-14-9-11-3-15-13-19-4-7-1-2-18-5-12-10-20
20000	27,44 horas	1297	15-11-17-6-14-3-8-4-16-9-19-1-13-2-7-5-18-12-10-20
50000	49,56 horas	1287	15-1-17-6-19-14-8-16-13-4-9-18-5-7-11-2-10-20-12-3

Tabla 11. Resultados de corridas del algoritmo genético propuesto
Fuente: Elaboración propia

A medida que se aumentó el número de iteraciones, el Makespan disminuyó hasta la corrida #3. A partir de la corrida # 3, el resultado no mejoró en términos de Makespan, si no que se mantuvo en el mismo valor. Sin embargo, con 50000 iteraciones se logró la mejor solución dada por el algoritmo genético propuesto y en comparación con los resultados dados en las instancias de Taillard(ta001-005) tenemos que:

- Límite inferior: 1232. La solución está alejada del límite inferior en sólo 4,46%
- Límite superior: 1278. La solución está alejada del límite superior en sólo 0,7%

Se puede concluir que, aunque el resultado no estuvo entre los límites encontrados por Taillard, no es tan significativa la diferencia. Además, no se hará énfasis en esta sección en mejorar este algoritmo debido a que el objetivo de esta investigación es la reprogramación de la producción para un ambiente tipo Flexible Flow Shop. Por otro lado, se espera que, en la programación para ambiente tipo Flexible Flow Shop, haya una mejoría en los resultados debido a que habrá más recursos de máquinas disponibles por etapa.

7. PROGRAMACIÓN DE LA PRODUCCIÓN EN AMBIENTE DE TIPO FLEXIBLE FLOW SHOP

El problema de programación de Flexible Flow Shop (FFS) consiste en n-trabajos que deben ser procesadas en varias etapas en serie, donde cada etapa está compuesta por m-máquinas generalmente idénticas en paralelo. La secuencia de todos los trabajos es el mismo durante todo el proceso, inicia en la Etapa 1 y finaliza en la Etapa k como se observa en la figura 10. Al igual que el problema tipo Flow Shop, debido a su complejidad este problema es categorizado como NP-Hard (Rahmani & Ramezanian, 2016).

El problema de FFS ha tenido gran relevancia en los últimos años debido a su pertinencia en los entornos reales de programación ya que la disponibilidad de más de una máquina dentro de cada etapa es una respuesta para eliminar las limitaciones por operaciones cuello de botella, por lo cual muchas directores de empresas con sistemas tipo Flow Shop clásico podrían estar

dispuestos a invertir en maquinaria que le permita mejorar su productividad (Keshavarz & Salmasi, 2013).

En FFS se tienen en cuenta las siguientes consideraciones (Morales, 2012) (Choi & Wang, 2012):

- Las máquinas en cada etapa son idénticas.
- Cada etapa E está compuesta por un conjunto de $M_E = \{1, \dots, m_E\}$ máquinas.
- La secuencia de N tareas tiene que ser procesado en cada etapa y en sólo una máquina.
- Una máquina puede procesar sólo un trabajo a la vez.
- La secuencia de los trabajos debe ser la misma para todo el proceso.
- Los trabajos tienen la misma opción de ser programados. No hay operaciones prioritarias.
- El procesamiento de un trabajo en una máquina no puede ser interrumpido.
- Se pueden procesar varias tareas simultáneamente en una misma etapa, siempre que se respete la secuencia.
- No existen relaciones de precedencia entre las operaciones de diferentes trabajos, pero sí existen relaciones de precedencia entre las diferentes operaciones de un mismo trabajo.
- El tiempo de procesamiento de cada tarea depende de la etapa.
- No se considera espera, sin embargo, se debe respetar la secuencia.

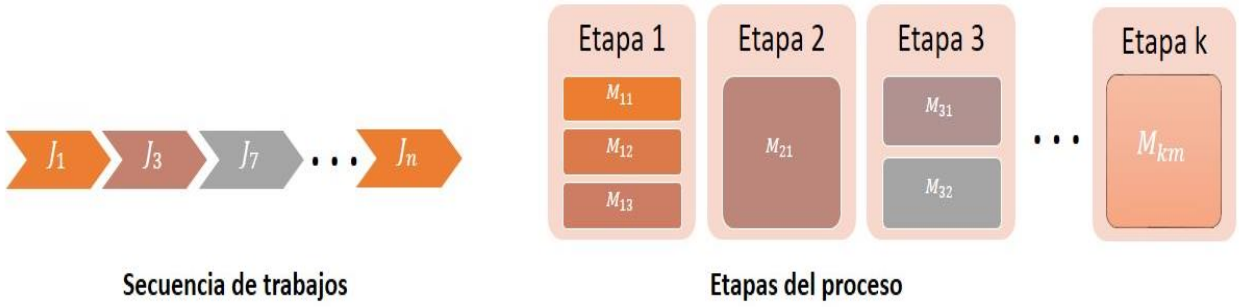

Figura 10. Flow Shop Flexible
Fuente: Elaboración propia

Los tipos de máquinas es un criterio que influye en el desempeño y en las limitaciones de optimización en un entorno de sistema de producción de tipo FFS. En la literatura, referente a FFS, más del 70% está enfocada en el estudio de este problema con máquinas en paralelo idénticas, el resto de las investigaciones están enfocadas en máquinas en paralelo uniformes y máquinas en paralelo no relacionadas. Por el lado de la función objetivo la mayoría de las investigaciones están relacionadas con el tiempo, donde la más común es la minimización del Makespan, el resto de las investigaciones están relacionadas con los trabajos y a funciones multiobjetivo (Lee & Loong, 2019).

7.1. Modelación matemática del problema de programación para sistemas tipo Flexible Flow shop

Para resolver el problema de tipo Flexible Flow Shop para minimizar Makespan donde no se permite la interrupción del procesamiento de un trabajo, el modelo matemático es el siguiente (Paternina, Montoya, Acero, & Herrera, 2008):

Notación:

j	Es el índice para identificar los trabajos, donde $j = 1, \dots, n$. $n = número\ total\ de\ trabajos$.
i	Es el índice para identificar las máquinas en una etapa s, donde $i = 1, \dots, m$.

s	Es el índice para identificar cada etapa, donde $s = 1, \dots, k$.
X_{jis}	Es una variable binaria que es igual a 1 si el trabajo j es asignado a la maquina i en la etapa s. De lo contrario, es igual a 0 (cero).
Y_{hjs}	Es una variable binaria que es igual a 1 si el trabajo h precede el trabajo j cuando se procesa en la etapa s. De lo contrario, es igual a 0 (cero).
R_j	Tiempo de liberación del trabajo j.
S_{js}	Tiempo de inicio del trabajo j en la etapa s.
C_{js}	Tiempo de finalización del trabajo j en la etapa s.
M	Es un número grande $M \to \infty$

El objetivo es minimizar Makespan:

$$\min C_{max} \ (7)$$

Sujeto a las siguientes restricciones:

$$C_{max} \geq C_{js}, \ para\ todo\ s = 1, \dots, k;\ j = 1, \dots, n \ \ (8)$$

La ecuación (8) asegura que el Makespan sea mayor o igual a los tiempos de finalización de cada trabajo.

$$C_{js} = S_{js} + p_{sj} \ \ (9)$$

La ecuación (9) es el cálculo del tiempo de finalización del trabajo j en la etapa s, donde p_{sj} es el tiempo de procesamiento del trabajo j en la etapa s.

$$\sum_{i=1}^{m_s} X_{jis} = 1, \ \ para\ todo\ s = 1, \dots, k;\ j = 1, \dots, n \ \ (10)$$

La ecuación (10) asegura que cada trabajo es asignado a una sola máquina en la etapa s.

$$\mathrm{C_{js}} \geq S_{j(s+1)}, \qquad para\ todo\ s\ =\ 1, \dots, k-1 \ \ (11)$$

Con la ecuación (11), se asegura que los trabajos inicien solo cuando se ha completado el trabajo en la etapa anterior.

$$S_{hs} \geq C_{js} - MY_{hjs},\ para\ todos\ los\ trabajos\ pares\ (h,j)\ (12)$$

El conjunto de restricciones (11) y (12) garantiza que solo haya un trabajo en una máquina de una etapa en un momento dado. Cuando $Y_{hjs} = 1$ y el trabajo h es anterior al trabajo j, se cumple la restricción en (12).

$$S_{js} \geq C_{hs} - (1 - M)Y_{hjs},\ para\ todos\ los\ trabajos\ pares\ (h,j)\ (13)$$

La ecuación (13) requiere que la hora de inicio del trabajo j en la etapa s sea posterior a la hora de finalización del trabajo h. Cuando $Y_{hjs} = 0$, lo que indica que el trabajo j es anterior al trabajo h, la restricción en 13 se cumple trivialmente y el tiempo de inicio del trabajo h en la etapa s debe ser el tiempo de finalización del trabajo j en la etapa s para satisfacer (12).

$$\mathrm{S_{j1}} \geq R_j\ (14)$$

Con la ecuación (14), los tiempos de inicio de los trabajos deben ser mayores a igual a los tiempos de liberación de los trabajos en el sistema.

$$X_{jis} \in \{0,1\}, \qquad Y_{hjs} \in \{0,1\}\ para\ todo\ s = 1,\dots,k;\ j = 1,\dots,n; y\ todo\ = 1,\dots,,m\ (15)$$

Finalmente, la ecuación (15) asegura que X_{jis} y Y_{hjs} asuman valores binarios 0 ó 1.

7.2. Esquemas de solución para el problema de programación para sistemas de tipo Flexible Flow shop

- **Branch and bound (B&B)**

El algoritmo de Branch and Bound se puede considerar el método más exacto para resolver el problema de FFS. En este algoritmo el conjunto de soluciones se define como la raíz del árbol, mientras que los subconjuntos del conjunto de soluciones están representados por las ramas del árbol. El algoritmo explora las ramas y determina un límite superior e inferior para la solución óptima. Con cada iteración el algoritmo descarta la solución hasta que se obtiene la solución óptima (Lee & Loong, 2019).

7.3. Desarrollo del algoritmo genético de Chu-Beasley para el problema de programación de máquinas de tipo Flexible Flow Shop

Para el desarrollo de este algoritmo se partirá del algoritmo genético propuesto para Flow Shop anteriormente. Para ello, se utilizarán los mismos datos de tiempo de procesamiento para cada trabajo de las instancias de Taillard (ta001-005) y cada etapa contará con 2 ó 3 máquinas de esta forma:

Etapa	#Máq
Etapa 1	2
Etapa 2	3
Etapa 3	3
Etapa 4	2
Etapa 5	2

Tabla 12. Número de máquinas por etapas
Fuente: Elaboración propia

Para este algoritmo se modificó la función para calcular el Makespan teniendo en cuenta que ahora son 5 etapas con cantidad de máquinas idénticas en paralelo como lo muestra la tabla 12.

El cálculo del Makespan se realizó programando cada trabajo por cada etapa de esta forma:

- Se programaron los dos primeros trabajos de la secuencia en todas las etapas, teniendo en cuenta que cada etapa tenía mínimo 2 máquinas por lo cual estos dos trabajos podían ser programados sin inconveniente. En la máquina 1 de cada etapa se programó el trabajo en la posición 1 de la secuencia y en la máquina 2 de cada etapa se programó el trabajo en la posición 2 de la secuencia.
- Luego, se programaron los trabajos desde la posición 3 hasta N en la etapa 1 partiendo de los dos primeros trabajos ya programados. Se programa cada trabajo en la máquina de la etapa 1 que se desocupa primero.
- Para la etapa 2 en adelante se programan los trabajos desde la posición 3 hasta N teniendo en cuenta en cada etapa que el trabajo se programa en la máquina que se desocupe primero de los trabajos anteriores. Luego, para el cálculo del Makespan se debe comparar el tiempo acumulado de cada trabajo en la etapa anterior con el tiempo acumulado de la máquina que se desocupa primero de los trabajos anteriores en la etapa actual, se escoge el mayor de estos y se le suma el tiempo de proceso del trabajo j $(j > 2)$ en la etapa actual.

Para este algoritmo propuesto de Flexible Flow Shop se mantuvo el tamaño de la población inicial $P = 500$ y el número de iteraciones en Tabu Search $W = 50$ constante para todas las corridas. La probabilidad de mutación se mantuvo en 0.4 en todas las corridas. Dentro del algoritmo genético se varía el número de iteraciones i donde está incluido Tabu Search, Cruzamiento y Mutación. A continuación, se presentan los resultados de cada corrida:

De los resultados anteriores se puede inferir que, aunque se aumente el número de iteraciones y por ende el tiempo de ejecución del algoritmo no hay una mejoría notoria en los resultados. Sin embargo, los resultados de cada corrida son muy similares y se puede destacar la solución con $i = 50000$ que arrojó un Makespan de 751.

i	*Tiempo*	Cmax	Secuencia
1000	18 minutos	754	17-14-11-7-6-15-4-19-9-1-5-3-2-16-8-10-18-12-20-13
2000	36 min	757	9-8-1-17-15-11-19-7-6-14-3-13-4-18-5-16-12-10-2-20
5000	1.5 horas	754	8-11-15-6-13-7-3-14-9-17-19-1-5-16-4-18-12-10-2-20
10000	3 horas	754	15-9-3-11-6-16-1-8-19-17-13-14-4-18-5-7-10-12-2-20
20000	6.1 horas	752	15-17-3-5-9-11-7-6-19-14-1-4-18-16-12-10-8-2-20-13
30000	9.14 horas	755	15-1-11-6-17-7-3-19-2-9-8-4-14-16-5-13-10-18-12-20
40000	12.31 horas	753	8-19-16-11-6-9-1-14-15-17-3-4-7-5-18-12-10-2-20-13
50000	15.47 horas	751	3-13-17-11-6-8-9-19-14-4-15-18-16-1-5-7-10-2-12-20

Tabla 13. Resultados para FFS usando las instancias de Taillard, con W (iteraciones de Tabu Search)= 50. Fuente: Elaboración propia

Con el fin de obtener mejores resultados, en el algoritmo se modificó el número de iteraciones dentro de Tabu Search con W=100. A continuación, se presentan los resultados:

I	*Tiempo*	Cmax	Secuencia
1000	35 minutos	761	15-17-1-6-14-3-4-9-16-13-19-8-5-18-12-10-11-7-2-20
2000	1.15 horas	758	14-17-6-8-3-15-19-16-9-1-4-13-18-7-11-12-5-2-10-20
5000	2.94 horas	755	14-9-11-6-3-4-15-5-1-17-8-19-18-16-2-10-13-7-12-20
10000	5.86 horas	752	17-15-1-9-6-4-5-11-16-8-14-19-18-10-12-7-2-3-20-13
20000	6.1 horas	751	15-11-4-9-6-19-1-17-14-8-3-7-5-18-16-12-10-2-20-13
30000	9.14 horas	753	11-19-17-3-14-8-15-9-5-6-7-18-16-1-4-13-12-10-2-20
40000	23.49 horas	750	15-7-9-11-6-14-3-17-1-19-4-5-8-16-18-12-10-2-20-13
50000	29.49 horas	752	11-3-16-9-15-6-7-8-17-19-4-1-14-5-18-12-10-2-20-13

Tabla 14. Resultados para FFS usando las instancias de Taillard, con W (iteraciones de Tabu Search) = 100.
Fuente: Elaboración propia.

En base a los resultados se puede observar que hasta $i = 10000$, hubo un poco de mejoría en los resultados. Sin embargo, aunque el resto de los resultados no fue mejorando a medida que

se aumentaba el número de iteraciones del algoritmo, se puede destacar que la mejor solución fue encontrada en $i = 40000$ con un Makespan de 750.

Para ambos casos, con $W = 50$ y $W = 100$, la mejor solución encontrada obtuvo un Makespan de 752. Por lo cual podemos inferir que, aunque se aumentó el número de iteraciones dentro de Tabu Search no hubo mejora significativa en los resultados.

8. Reprogramación de la producción bajo condiciones de incertidumbre

8.1. Programación de la producción ante diferentes tipos de incertidumbres

En el capítulo anterior, en el estado del arte, se identificaron diferentes tipos de incertidumbres que pueden afectar un sistema de producción y se resaltó la necesidad que la planeación de la producción se realice incluyendo estos eventos inesperados con el fin de mitigar su impacto y de cierta forma responder ante ellos cuando se presenten. Cuando la programación no tiene ningún tipo de control ante estas incertidumbres, todo lo que sucede en el futuro es una sorpresa, suceden eventos de manera inesperada sin ninguna política que permita reaccionar ante estos, por lo que no hay una mitigación activa frente a los efectos secundarios que suceden como producto de que ocurran dichos eventos. Sin embargo, en los esquemas de producción a menudo estas incertidumbres se descuidan ya que son difíciles de modelar y tener en cuenta algorítmicamente. Por lo general, los supuestos simplificadores se realizan para asumir que toda la información es determinista, estática en el tiempo y conocida de antemano (Larsen & Pranzo, 2019).

Una programación determinística se realiza con la esperanza que una vez inicie el proceso de producción este se va a desarrollar de manera satisfactoria según lo planificado. La programación estocástica por su parte incluye variables aleatorias que se pueden enfocar en políticas de control locales con el objetivo de minimizar alguna medida de desempeño. Sin embargo, muchos esquemas de producción, en la vida real deben enfrentarse a diferentes tipos de incertidumbres que impiden que la ejecución de la producción se desarrolle según el plan de producción inicial.

Pinedo identifico 12 diferencias principales entre los modelos teóricos de programación y los problemas reales que se presentan en la ejecución de un programa de producción, entre estos podemos encontrar los siguientes problemas: (Katragjini, Vallada, & Ruiz, 2013):

- Constantemente llegan nuevos trabajos al sistema.
- ***Muchos esquemas de producción no incluyen reprogramación.***
- Los entornos de las máquinas son más complicados.
- ***La prioridad de los trabajos puede cambiar en cualquier momento.***
- Las preferencias al momento de seleccionar una máquina son importantes.
- Existen restricciones en la disponibilidad de máquinas de acuerdo con los turnos de trabajo o los horarios.
- Presentan funciones de penalización no lineales.
- ***Los esquemas de programación, a menudo, deben considerar más de un objetivo.***
- Se puede influir en la capacidad disponible (carga de trabajo, turnos de trabajo).
- Los tiempos de procesamiento pueden no seguir ninguna distribución estadística.
- Los tiempos de procesamiento en la misma máquina tienden a tener una correlación muy positiva.
- Las distribuciones del tiempo de procesamiento pueden variar debido a cambios por el aprendizaje de los operarios o del mismo staff, o al deterioro de las máquinas.

De estos 12 problemas descritos se pueden destacar tres (ver ítems resaltados). Se destaca nuevamente la importancia de la reprogramación con el fin de dar mejor respuesta ante las incertidumbres que se puedan llegar a presentar, además debido a la importancia de redirigir los trabajos a causa de un elemento disruptivo, teniendo en cuenta a cuál de estos trabajos se le debe dar prioridad, respetando los tiempos de entrega; y por último se resalta la importancia de que en la programación se considere más de un objetivo como sucede en la vida real, en el objetivo de esta investigación incluimos las métricas de desempeño de Makespan y tardanza total del programa de producción asociado.

La variabilidad en los tiempos de procesamiento, por ejemplo, tiene diferente impacto si ocurre al inicio de la jornada o al final de la jornada. La incertidumbre por daños de materiales puede tener también diferente impacto en la métrica de desempeño cuando tiene en cuenta tiempos de entrega, si ocurre luego de varias operaciones en las que ya se ha invertido, materia prima, mano de obra, entre otros, a que si ocurre en la primera operación. La incertidumbre que afecta la disponibilidad de materiales podría ser más importante que los impactos del tiempo, a veces, a veces no. Existen tres dimensiones claves que pueden ayudar a categorizar las formulaciones de los problemas (Aytug, Lawley, McKay, Mohan, & Uzsoy, 2005):

- Causa, puede ser un objeto como el material, una herramienta, el personal, etc.; y el estado como listo, no listo, calidad del material, etc.
- Contexto, se refiere a la situación del sistema en el momento del evento programado, y se clasifica en libre o sensible. Cuando una situación es libre de contexto no necesita ninguna información adicional o alguna toma de decisiones especial. Por otro lado, cuando es sensible al contexto si requiere información sobre el contexto y las implicaciones asociadas; sí hay algo en el contexto que pueda afectar alguna métrica de rendimiento como, por ejemplo, la experiencia del operario.
- Impacto, se refiere a la consecuencia que trae consigo el evento inesperado. Por ejemplo, si esta incertidumbre afecta el tiempo de procesamiento, si afecta uno o varios recursos o si afecta algún costo asociado al sistema, entre otros.

8.2. Esquemas de solución para el problema de reprogramación de la producción bajo condiciones de incertidumbre

En un entorno real de un sistema de producción se pueden presentar interrupciones o eventos inesperados dentro del esquema de programación que pueden afectar la consecución de los objetivos inicialmente propuestos. Por ello, se hace necesario incluir reprogramación dentro de estos esquemas de producción como una manera práctica de responder ante eventos inesperados de forma que no se vean tan afectados los objetivos propuestos por las empresas. Entre los tipos más comunes de interrupciones que crean la necesidad de reprogramar se

encuentran: cambios en las fechas de lanzamiento, llegada de nuevos pedidos, cancelaciones de pedidos, cambios en la prioridad de los pedidos, escasez de materiales, retrasos en el procesamiento y averías de la máquina (Liu & Ro, 2014) & (Hall & Potts, 2010).

En la reprogramación se debe ajustar un cronograma previamente planificado que se ve afectado debido a alguna interrupción presentada dentro de la ejecución de la programación. Por lo cual, la reprogramación de la producción debe procurar en lo posible encontrar un nuevo programa rentable, que sea cercano a las metas propuestas en la programación inicial, de manera que evite cambios excesivos de esta programación inicial (Hall & Potts, 2010).

Muchas investigaciones sobre programación de la producción se enfocan en resolver problemas en ambientes estáticos sin considerar que un entorno real de programación es dinámico y está sujeto a una amplia gama de incertidumbres y para el caso del problema de programación de la producción para ambientes tipo Flexible Flow Shop se ha demostrado que es de tipo NP hard y que éste es particularmente difícil de resolver (Choi & Wang, 2012) debido a que es un proceso complicado de toma de decisiones ya que compromete un número de trabajos que deben ser programados en un conjunto de máquinas distribuidas en varias etapas.

En la literatura se pueden identificar tres tipos de métodos para resolver el problema de Flexible Flow Shop: métodos de programación matemática, heurísticas y metaheurísticas. Sin embargo, aunque existen muchas investigaciones sobre programación de la producción para FFS, no es igual para el caso de reprogramación de FFS, donde la mayoría de las investigaciones ha utilizado metaheurísticas en sus métodos de solución para abordar este problema (Peng, y otros, 2019).

A continuación, se presentarán algunas investigaciones sobre cómo han abordado la problemática de reprogramación para Flexible Flow Shop (FFS):

- **Tadeusz Sawik (2007)**

Sawik propuso nuevos algoritmos basados en formulaciones de programación de enteros mixtos para la reprogramación de la producción en ambientes de manufactura dinámicos de fabricación por encargo (Make to order), donde los clientes pueden modificar, cancelar o agregar pedidos durante el horizonte de planificación.

Esta investigación está enfocada en la problemática de programación para FFS con máquinas idénticas en paralelo donde el horizonte de programación se realiza con un set J de órdenes de los clientes que tienen en cuenta 3 parámetros: 1) El periodo más temprano de disponibilidad del material, 2) Fecha de envío requerida por el cliente y 3) El tamaño de la orden.

Cada orden es procesada por cada etapa, sin embargo, se considera que algunas órdenes puedan pasar por alto algunas etapas. El tiempo del proceso depende de cada orden en cada etapa según el tipo de producto solicitado. Las órdenes son procesadas en cada etapa por lotes en tamaños Small-size y Large-size según los tipos de productos solicitados.

El objetivo de la programación reactiva a largo plazo es asignar o reasignar pedidos de clientes en períodos de planificación a lo largo de un horizonte de planificación para maximizar el nivel de servicio al cliente (minimizando el número de órdenes retrasadas) a la vez que limita el inventario de entrada y de salida. El horizonte de planificación consta de h períodos de planificación (por ejemplo, días laborales).

Para resolver este problema Sawik propone varias políticas de reprogramación, desde una reprogramación de las órdenes restantes de los clientes o decidir no hacerlo. Se establece t_{mod} como el primer período de planificación inmediatamente después de la modificación de la orden. Se supone que los pedidos de clientes completados antes de t_{mod} o con fechas de vencimiento inferiores a t_{mod} no se pueden modificar. El algoritmo REALL permite la reprogramación de todos los pedidos que han sido asignados a períodos no inferiores a t_{mod}, mientras que el algoritmo RENON la asignación de todas las órdenes restantes, no inferiores a t_{mod}, no cambia. Además, se considera una política de restricción media denominada algoritmo REMAT para reprogramar

solo un subconjunto de los pedidos de clientes restantes que están a la espera de suministros de materiales. La selección de alguno de estos algoritmos depende del impacto que causen las interrupciones.

- **S. H. Choi, & K. Wang (2012)**

En su investigación estos autores proponen un nuevo enfoque basado en la descomposición (DBA), el cual combina la regla de tiempo de procesamiento más corto (SPT) y el algoritmo genético, cuyo objetivo es minimizar el Makespan en un FFS con tiempos de procesamiento estocásticos.

Este problema se aborda en un entorno de FFS con máquinas paralelas idénticas, donde los trabajos son procesados en cada etapa que tiene m máquinas. El tiempo estocástico del proceso se calcula con la suma del tiempo esperado del proceso y la desviación estándar. La incertidumbre analizada en esta investigación es el coeficiente de la variación de los tiempos de proceso CVTP, que se calcula por la relación entre la desviación estándar y el tiempo de procesamiento esperado. Esta incertidumbre puede ser ocasionada por problemas de calidad, desgaste de la herramienta, disponibilidad del operador, entre otros.

Para resolver este problema estos autores proponen lo siguiente:

- Descomponer el FFS en varios clústeres de máquinas que tienen en común una naturaleza estocástica similar. Esta naturaleza se determina con el uso del CVTP que representa la incertidumbre de los tiempos de procesamiento.
- Asignar un enfoque a cada clúster de máquinas, estos enfoques pueden ser uno de carácter predictivo-reactivo o completamente reactivo.
- Generar e integrar subprogramas de producción de los clústeres de máquinas.

Debido a que agrupar las máquinas en clústeres no se puede hacer de manera tradicional, los autores propusieron un algoritmo de vecindad K-means para descomponer el FFS teniendo en cuenta: 1) cómo agrupar las máquinas dentro de un clúster con el cálculo de la diferencia de la naturaleza estocástica entre dos estaciones de máquinas paralelas computando las distancias de sus vectores estocásticos, y 2) como escoger un apropiado número de clúster de máquinas.

Luego de obtener los clústeres se le asigna un enfoque completamente reactivo con tiempo de proceso más corto (SPT) o un enfoque predictivo-reactivo con algoritmo genético (AG). Para elegir cual enfoque es más adecuado para cada clúster se calcula la diferencia del Makespan de las dos programaciones generadas tanto con SPT como con AG con la fórmula:

$$MDSG = \frac{M_{spt} - M_{GA}}{M_{GA}} \quad (16)$$

Si el resultado de $MDSG$ es positivo indica que las máquinas tienen naturaleza estocástica baja por lo que se debe utilizar el algoritmo genético. Si, por el contrario, es negativo, indica que el Makespan calculado con SPT es menor que el del AG y que las máquinas tienen una naturaleza estocástica alta.

- **Donya Rahmani y Reza Ramezanian (2016)**

Esta investigación está enfocada en un problema de reprogramación en un entorno de FFS dinámico que considera llegadas inesperadas de nuevos trabajos en el proceso como interrupciones y para resolverlo proponen un nuevo modelo de búsqueda de vecindario variable.

Se realiza una programación inicial con un modelo matemático determinístico con los trabajos disponibles en el sistema y con la información respectiva de tiempos de proceso y fecha de entrega con el fin de minimizar la tardanza ponderada total. Los nuevos trabajos que llegan no se tratan como prioritarios. Para que el sistema de producción pueda continuar cuando se

presenta un nuevo trabajo, se usa un modelo de reprogramación llamado SWT que además del objetivo inicial tiene en cuenta dos medidas: estabilidad y resistencia al cambio.

La estabilidad es una medida que indica el grado de reordenamiento de los trabajos después de la reprogramación, con la cual se calcula la diferencia entre el tiempo de finalización de los trabajos de la programación inicial y la nueva programación obtenida luego de la interrupción. Si el nuevo trabajo que llega se programa al final de la secuencia, el tiempo de inicio de los trabajos será igual al de la secuencia inicial, por lo cual esta nueva programación se considera estable. Por su parte, la medida de resistencia al cambio es un problema conocido en la literatura para las industrias que consiste en la resistencia que pueden mostrar los directores de producción o los trabajadores ante cambios en el programa de producción debidos a las interrupciones, como resultado de esta resistencia se puede ver afectado el tiempo de procesamiento con el aumento de estos.

Debido a la complejidad computacional para resolver este problema los autores proponen un algoritmo de búsqueda con vecindario variable (VNS). Este algoritmo tiene dos funciones de sacudida (shaking), que permite buscar soluciones en un vecindario vecino; y una función de búsqueda local que permite explorar y mejorar las soluciones dentro de un vecindario local.

- **Kunkun Peng, Quan-Ke Pan, Liang Gao, Xinyu Li, Swagatam Das, Biao Zhang (2019)**

En esta investigación los autores tratan el problema de Flow Shop Hibrido (HFS) que consiste en n trabajos que deben ser procesados en m etapas, donde por lo menos una etapa tiene 2 o más máquinas en paralelo. Cada trabajo debe procesarse de manera secuencial desde la etapa 1 hasta la i-esima etapa. La ruptura en este problema de producción esta ocasionada por alguno de estos 3 eventos dinámicos: averías de máquinas, llegada de nuevos trabajos o variación en los tiempos de liberación de los trabajos. Debido a la ruptura ocasionada por alguno de estos 3 eventos la programación inicial no es factible, por lo cual se debe generar una reprogramación de la producción.

El algoritmo propuesto por estos autores tiene como objetivo minimizar el Makespan y la inestabilidad del sistema, para ello, este enfoque de programación busca calcular límites superior e inferior para ambos objetivos de optimización. Mientras que el Makespan es el tiempo de finalización de todos los trabajos, la inestabilidad del sistema se puede medir por el número de operaciones o trabajos que sufren modificación en el tiempo de inicio de sus trabajos comparando el horario de la reprogramación, luego de una ruptura, con el horario de la programación inicial del sistema.

Para resolver este problema los autores proponen un algoritmo de descenso de vecindario variable de arranque múltiple (MSVND) para la reprogramación del HFS. El algoritmo MSVND está basado en el básico VND, se desarrolla una decodificación híbrida donde existe una combinación de la decisión de la secuencia de procesamiento de trabajos sin terminar y la asignación de la máquina. Para mejorar la intensificación del MSVND, se utiliza un algoritmo de búsqueda local que está basado en el algoritmo de optimización de la mosca de la fruta (FFO) y un algoritmo de búsqueda local mejorado basado en FFO, con el objetivo de encontrar la mejor solución o el óptimo del problema. Además, se introduce un criterio de aceptación para determinar si pueden ser aceptados los óptimos locales y una estrategia de reinicio con perturbación para realizar la búsqueda en zonas no exploradas.

8.3. Desarrollo de un algoritmo genético para el problema de programación y reprogramación de la producción bajo condiciones de incertidumbre

Para lograr el propósito de esta investigación se hace necesario diseñar primero un modelo programación de tipo Flexible Flow Shop con el fin de obtener la secuencia inicial del programa de producción, cuyo objetivo es minimizar el Makespan y la tardanza del programa asociado. Este modelo parte del expuesto anteriormente basado en el algoritmo genético de Chu-Beasley, con algunas modificaciones para el cálculo del fitness, teniendo en cuenta que hay varias máquinas por etapa, y cambios en las funciones de cruzamiento y mutación para que se realicen de manera aleatoria. Este algoritmo tiene en cuenta el modelo matemático expuesto en la sección 7.1 pero

en este caso los trabajos pueden tener tiempos de espera entre las etapas, por lo cual no se tiene en cuenta algunas restricciones del modelo.

Este modelo de programación será comparado con modelos similares encontrados en la literatura con enfoque en el tipo de problema de asignación de máquina para Flexible Flow Shop que tenga como objetivo la minimización del Makespan o la tardanza total, o en lo posible ambos.

8.4. Desarrollo del modelo de programación de la producción para el problema de programación de máquinas de tipo Flexible Flow Shop.

Este algoritmo resuelve un problema bi-objetivo para minimizar Makespan y tardanza total. A continuación, se definen las variables utilizadas en este algoritmo y se describe su funcionamiento:

- ME: # de máquinas por etapa
- N: número de trabajos según la información del problema
- P: tamaño de las dos poblaciones iniciales
- W_{ag}: # de iteraciones del algoritmo genético
- W_{ts}: # de iteraciones para Tabu Search
- W_1 y W_2: peso dado al Makespan y a la tardanza
- R: Número aleatorio entre 0 y 1.
- ProbM: probabilidad de mutación

Paso 1. Se ingresa un archivo con la información concerniente a los tiempos de procesamiento de cada trabajo por etapa y los tiempos máximos de entrega de cada trabajo.

Paso 2. Se modifican las variables ME, W_{ag}, W_{ts}, W_1, W_2, ProbM y P. Luego de realizar los dos primeros pasos se puede ejecutar el algoritmo.

Paso 3. El algoritmo comienza generando dos poblaciones que generan P secuencias. A estas secuencias se les calcula la función fitness para determinar los valores de Makespan y tardanza de cada una.

Paso 4. Se utiliza una función para normalizar el fitness de las secuencias obtenidas del paso 3 para cada población y de esta manera escoger la mejor secuencia de cada una de ellas según los pesos W_1 y W_2 determinados antes de ejecutar el algoritmo. Se obtienen dos secuencias, una de cada población, y se convierten en las secuencias iniciales para iniciar Tabu Search.

Paso 5. Se realiza Tabu Search intercambiando dos trabajos para cada una de las dos secuencias obtenidas en el paso 4. La lista tabú tiene un tamaño de 3 movimientos. Tabu Search se realiza hasta cumplir las W_{ts} iteraciones determinadas al inicio. Se escoge la secuencia con mejores resultados en términos de minimización de Makespan y tardanza según el cálculo de fitness y la función de normalización.

Paso 6. Del paso anterior se obtiene el padre 1 y el padre 2 para dar inicio al cruzamiento. En el cruzamiento se inserta una fracción de un padre al otro, a su vez se identifican los trabajos repetidos y se cambian por los trabajos faltantes logrando que sea factible. Se obtienen dos nuevas secuencias denominadas hijo 1 e hijo 2, se elige el hijo que presente mejores resultados en términos de minimización Makespan y tardanza según el cálculo de fitness y la función de normalización.

Paso 7. La secuencia hijo puede ser sometida a mutación de acuerdo con una probabilidad determinada, intercambiando trabajos entre la secuencia. Si la secuencia obtenida por la mutación es mejor que la secuencia hijo, se reemplaza la secuencia hijo por la nueva solución.

Paso 8. Del paso 5 al 7 se repiten hasta cumplir las W_{ag} iteraciones. El algoritmo genera una matriz con todas las mejores secuencias obtenidas de cada iteración. Se calcula el fitness de cada secuencia y usando la fórmula de normalización se escoge la mejor secuencia con el menor Makespan y tardanza según los pesos W_1 y W_2. Esta secuencia es la solución final del algoritmo.

Paso 9. Por último, se obtiene una matriz con los tiempos de finalización de cada trabajo en cada etapa correspondientes a la secuencia final.

8.4.1. Funciones creadas en MATLAB para la ejecución de este algoritmo

A continuación, se explicarán a detalle varias funciones creadas para la ejecución de este algoritmo en MATLAB:

- **Generar población aleatoria**

Según el número de trabajos del problema se crea una matriz con un numero P de secuencias de tamaño N que se generan de manera aleatoria usando una función de permutación de MATLAB.

- **Normalización para funciones con más de un objetivo.**

Debido a que esta investigación tiene un problema multiobjetivo que para el caso de la programación de la producción es minimizar Makespan y tardanza total, y en el caso de la reprogramación es minimizar la diferencia entre los tiempos de inicio de la programación original y la nueva obtenida con la reprogramación y la tardanza total. Uno de los métodos más conocidos para resolver problemas multiobjetivo es la función de utilidad aditiva ponderada (Dai, Tang, Giret, Salido, & Li, 2013):

$$U(k) = W_1 f_{1k} + W_2 f_{2k} + \cdots + W_i f_{ik} \quad (17)$$

Donde f es la función con k-tuplas de objetivos y $W_1, W_2, \dots, W_k$ son los pesos de importancia de cada función objetivo. Se requiere que la sumatoria de estos pesos sea igual a 1. Los valores de W los define quien esté a cargo de la toma de decisiones, que en este caso puede ser quien lidera el proceso de producción. Cuando es difícil asignar estos pesos por que las funciones objetivo están en diferentes escalas se utiliza un enfoque simple para normalizar los valores de los criterios objetivos (Malakooti, 2011). Donde f_1', f_2', f_k' , son los valores normalizados para f_{1k}, $f_{2k}, y\ f_{ik}$ y se calculan con la siguiente formula:

$$f_{ik}' = \frac{f_{ik},max - f_{ik}}{f_{ik},max - f_{ik},min} \quad (18)$$

En esta fórmula los valores de f_{ik}, max y f_{ik}, min, corresponden a los valores máximo y mínimo de la función objetivo en cuestión.

En el desarrollo de este algoritmo se incluye esta fórmula de normalización, cada vez que se deba escoger la mejor secuencia entre las diferentes opciones que se presentarán en varias etapas del algoritmo, buscando la secuencia con menor Makespan y tardanza total según los pesos W1 y W2 determinados según el caso. Se pueden presentar 3 casos para escoger la mejor secuencia:

1. Si las secuencias finales tienen la misma tardanza, seta escoge la de menor Makespan, si hay más de dos con el mejor resultado, se elige una secuencia de manera aleatoria.
2. Si las secuencias finales tienen el mismo Makespan, se escoge la de menor tardanza, si hay más de dos con el mejor resultado, se elige una secuencia de manera aleatoria
3. En caso de que las secuencias tengan diferentes fitness, se utiliza la función de normalización para funciones multiobjetivo, los pesos w1 y w2 son determinados por quién maneja la programación

- **Cálculo del fitness para Makespan y tardanza**

- Se reordena los tiempos de procesamientos según la nueva secuencia.
- Se programa el trabajo 1 según la secuencia en todas las etapas. En cada etapa se va sumando el tiempo acumulado de la etapa anterior con el tiempo de proceso en esa etapa según el trabajo 1.
- Se programa el resto de los trabajos en la etapa 1. Los trabajos se asignan en esta etapa en la máquina que se desocupe primero.
- Luego se puede programar todos los trabajos diferentes a 1 en el resto de las etapas, teniendo en cuenta que el trabajo se ubica en cada etapa en la máquina que se libera más pronto. Luego, para el cálculo del Makespan se debe comparar el tiempo acumulado de cada trabajo en la etapa anterior con el tiempo acumulado de la máquina que se desocupa primero de los trabajos anteriores en la etapa actual, se escoge el mayor de estos y se le suma el tiempo de proceso del trabajo j $(j > 2)$ en la etapa actual.

Al obtener los tiempos acumulados en cada etapa por trabajo, se obtiene el tiempo final de cada trabajo en la última etapa. El Makespan corresponde al trabajo con mayor tiempo de finalización. Por su parte, para el cálculo de la tardanza, se calcula el valor absoluto de la diferencia entre el tiempo final de cada trabajo en la última etapa con el tiempo de entrega máximo al cliente del trabajo correspondiente, la tardanza total será la sumatoria de todas estas tardanzas.

- **Tabu Search**

Este esquema es igual al utilizado en el modelo de programación. Con esta función se evalúan por separado dos secuencias que parten de la secuencia inicial:

Padre inicial 1: Es la misma secuencia inicial desde Bd hasta N

Padre inicial 2: Partiendo del padre inicial 1, se traslada el trabajo donde ocurre la interrupción de la programación hasta la posición final N.

Se inicia con la secuencia **Padre inicial 1** y se realiza el mismo procedimiento explicado en el modelo de programación. La diferencia para el modelo de reprogramación, es que la función de normalización permite escoger la mejor secuencia según la que presente los mejores resultados en términos de minimizar la sumatoria de la diferencia de los tiempos iniciales y la tardanza. Como resultado, la mejor secuencia obtenida se convierte en el Padre 1 con el cual se realizará el cruzamiento. Ejemplo para 20 trabajos con Bd=12:

Posición	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Padre 1	14	20	15	6	3	1	7	2	12	8	10	19	16	13	9	4	18	5	17	11
Sec Tabu S.	14	20	15	6	3	1	13	2	12	8	10	19	16	7	9	4	18	5	17	11

Figura 11. Representación de una nueva secuencia obtenida con Tabu Search.
Fuente: Elaboración propia.

Se actualiza lista tabú con los siguientes resultados: $Lista\ tab\acute{u} = \{7\ 14;\ 2\ \ 4; 6\ \ 9\}$

Para obtener el padre 2, necesario para el cruzamiento, se aplica Tabu Search partiendo de la secuencia del **Padre inicial 2**.

- **Cruzamiento del padre 1 y padre 2**

  - Se inicia con las dos mejores secuencias obtenidas por la función de Tabu Search como se explicó anteriormente, es decir el padre 1 y el padre2.
  - Se elige de manera aleatoria una parte del cromosoma, su tamaño es hasta máximo la mitad del cromosoma, desde una posición i hasta una posición j, donde $i > 1$ y $j < n$.
  - Se inserta la fracción del padre 2 en el padre 1, a su vez se identifican los trabajos repetidos y se cambian por los trabajos faltantes del padre 1 respetando el orden en que se encontraban en la secuencia inicial para lograr que sea factible. Luego se inserta la fracción del padre 2 en el padre 1 y se procede de la misma manera que el anterior. De esta forma se obtienen 2 hijos. Ejemplo para una secuencia de 20 trabajos donde la fracción a cruzar en los padres corresponde a la posición 5 hasta la 8:

Posición	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Padre 1	19	20	5	4	3	1	7	2	6	18	17	11	13	8	14	9	12	15	10	16
Padre 2	14	1	3	6	7	20	15	12	2	8	10	19	16	13	9	4	18	5	17	11
Padre 1X	19	*	5	4	7	20	15	12	6	18	17	11	13	8	14	9	*	*	10	16
Padre 2X	14	*	*	6	3	1	7	2	*	8	10	19	16	13	9	4	18	5	17	11
Hijo 1	19	**3**	5	4	7	20	15	12	6	18	17	11	13	8	14	9	**1**	**2**	10	16
Hijo 2	14	**20**	**15**	6	3	1	7	2	**12**	8	10	19	16	13	9	4	18	5	17	11

Figura 12. Representación entre el cruzamiento del padre 1 y padre 2.
Fuente: Elaboración propia

  - A los dos hijos resultantes se les calcula el fitness y se utiliza la fórmula de normalización para elegir la secuencia hijo con mejores resultados para Makespan y tardanza.

- **Mutación**

La ejecución de esta función está sujeta a una probabilidad determinada antes de ejecutar el algoritmo llamada ProbM. Se elige una fracción aleatoria de la secuencia hijo de manera similar al cruzamiento.

En esta fracción se combinan los trabajos ubicados en esas posiciones de manera aleatoria. Ejemplo en una secuencia de 20 trabajos intercambiando los trabajos desde la posición 9 hasta la 14:

Posición	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Hijo 2	14	20	15	6	3	1	7	2	12	8	10	19	16	13	9	4	18	5	17	11
Hijo M	14	20	15	6	3	1	7	2	19	12	16	13	10	8	9	4	18	5	17	11

Figura 13. Representación del proceso de mutación para la secuencia hijo.
Fuente: Elaboración propia.

Si la secuencia obtenida de la mutación tiene un mejor resultado en términos de Makespan y tardanza que la secuencia hijo, se escoge esta nueva secuencia, de lo contrario, permanece la secuencia hijo como mejor solución. Para escoger entre estas dos secuencias se hace necesario utilizar la función de normalización.

8.5. Desarrollo del modelo de reprogramación de la producción para el problema de programación de máquinas de tipo Flexible Flow Shop

El presente algoritmo permite actualizar la programación de la producción obteniendo una nueva secuencia luego de que ocurre una interrupción debido a la no disponibilidad de materia prima que afecta la ejecución del plan de producción en curso. Con esta nueva secuencia se espera obtener resultados similares de Makespan y tardanza con respecto a la solución inicial. Por lo cual, este algoritmo es bi-objetivo, pero a diferencia del modelo de programación, este busca minimizar la sumatoria de la diferencia entre los tiempos de inicio de la programación inicial y la nueva programación y minimizar la tardanza total. Se espera que al minimizar la diferencia de los tiempos de inicio, esta nueva secuencia de programación sea muy parecida a la secuencia inicial y que sumado a la minimización de la tardanza, se logre obtener resultados de Makespan muy similares al de la solución inicial.

Este modelo de reprogramación se desarrolló a partir del modelo de programación descrito anteriormente, realizando los ajustes correspondientes para calcular correctamente los tiempos

de finalización de cada trabajo por etapa, teniendo en cuenta el trabajo donde se produjo la interrupción y su nuevo tiempo de liberación, además se agregó una nueva función denominada 2 OPT con el objetivo de mejorar la solución hijo. El desarrollo de este algoritmo consta de los siguientes pasos:

Paso 1. Se ingresa un archivo de Excel que contenga la información concerniente a la programación inicial: los tiempos de procesamiento de cada trabajo por etapa, los tiempos máximos de entrega de cada trabajo, los tiempos de finalización de cada trabajo por etapa y los tiempos de liberación de cada trabajo.

Paso 2. Se modifican las variables:

- ME: # de máquinas por etapa
- SecProg: secuencia de la programación inicial
- CmaxTmaxSecProg: makespan y tardanza de la programación inicial
- W_{ag}: # de iteraciones del algoritmo genético
- W_{ts}: # de iteraciones para Tabu Search
- W_1 y W_2: peso dado al Makespan y a la tardanza
- ProbM: probabilidad de mutación
- B1: posición del trabajo que se reprogramo anteriormente, si no lo hay, se coloca cero (0)

Paso 3. De acuerdo con los tiempos de procesamiento de cada trabajo por etapa y los tiempos de finalización de cada trabajo por etapa se procede a calcular el tiempo de inicio de cada trabajo según la secuencia de programación inicial en la etapa 1.

Paso 4. De manera aleatoria se obtiene una posición Bd de la secuencia que debe ser mayor a B1, donde ocurrirá la interrupción. El tiempo de inicio para el trabajo ubicado en esta posición es un número aleatorio entre el tiempo inicial de este trabajo en la secuencia de la programación inicial y el valor de Makespan de la misma.

Paso 5. El algoritmo genera dos secuencias que parten de la secuencia inicial: 1) Es la misma secuencia inicial desde Bd hasta N y 2) Partiendo de la secuencia 1, se traslada el trabajo donde ocurre la interrupción de la programación hasta la posición final (N).

Paso 6. Se realiza Tabu Search para cada una de las dos secuencias obtenidas en el paso anterior. Se intercambian 2 trabajos en las secuencias y la lista tabú tiene un tamaño de 3 movimientos. Tabu Search se realiza hasta cumplir las W_{ts} iteraciones determinadas al inicio. Se escoge la secuencia con el mejor resultado en minimización de diferencia de tiempo inicial y tardanza para cada caso.

Paso 7. Del paso anterior se obtiene el padre 1 y el padre 2 para dar inicio al cruzamiento. En el cruzamiento se inserta una fracción de un padre al otro, a su vez se identifican los trabajos repetidos y se cambian por los trabajos faltantes logrando que sea factible. Se obtienen dos nuevas secuencias denominadas hijo 1 e hijo 2, se elige el hijo que presente mejores resultados en términos de minimización de la diferencia de los tiempos de inicio y tardanza según el cálculo de fitness y la función de normalización.

Paso 8. La secuencia hijo puede ser sometida a mutación de acuerdo con una probabilidad determinada, intercambiando algunos trabajos entre la secuencia. Si la secuencia obtenida por la mutación es mejor que la secuencia hijo, se reemplaza la secuencia hijo por la nueva solución.

Paso 9. Para finalizar, la secuencia obtenida en el paso anterior se somete a una función denominada 2 OPT con el objetivo de intercambiar dos posiciones adyacentes y de esta forma buscar una mejor solución. Si la secuencia obtenida usando 2 OPT tiene un mejor fitness, se escoge esta, de lo contrario, permanece la anterior.

Paso 10. Del paso 6 al 9 se repiten hasta cumplir las W_{ag} iteraciones. El algoritmo genera una matriz con todas las mejores secuencias obtenidas de cada iteración. Se escoge la secuencia con mejor fitness según los pesos W_1 y W_2. Esta secuencia es la solución final del algoritmo. Por último, se obtiene una matriz con los nuevos tiempos de finalización de cada trabajo en cada etapa correspondientes a la secuencia final.

8.5.1. Funciones creadas en MATLAB para la ejecución de este algoritmo

La mayoría de estas funciones parten de las creadas en el modelo de programación, pero con algunas modificaciones necesarias para el caso de la reprogramación. En cada una de estas se conserva de la secuencia obtenida en la programación de la producción el orden de los trabajos

desde 1 hasta Bd-1, es decir, hasta donde ocurre la interrupción. Ejemplo, una secuencia de 20 trabajos con Bd=12:

Posición	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Sec Prog	14	20	15	6	3	1	7	2	12	8	10	19	16	13	9	4	18	5	17	11

Reprog	14	20	15	6	3	1	7	2	12	8	10	19	16	13	9	4	18	5	17	11

Figura 14. Representación secuencia de reprogramación.
Fuente: Elaboración propia.

En este caso, en la nueva secuencia deben permanecer fijos los trabajos ubicados desde la posición 1 hasta la 11 y los trabajos a reprogramar van desde la posición 12 hasta la 20 o N.

- **Secuencias iniciales**

El algoritmo genera dos secuencias que parten de la secuencia inicial: 1) Es la misma secuencia inicial desde Bd hasta N y 2) Partiendo de la secuencia 1, se traslada el trabajo donde ocurre la interrupción de la programación hasta la posición final (N).

- **Normalización para funciones con más de un objetivo.**

Esta función se utilizará del mismo modo que el expuesto en el modelo de programación, pero en este caso los dos objetivos a minimizar son:

- La sumatoria de la diferencia entre los tiempos de inicio de la programación inicial y la nueva secuencia generada por el modelo de reprogramación.
- La tardanza total.

Los pesos W_1 y W_2 son determinados por el programador y corresponden a la diferencia del tiempo inicial y la tardanza respectivamente. Se pueden presentar 3 casos para escoger la mejor secuencia:

1. Si las secuencias finales tienen la misma tardanza, se escoge la de menor diferencia de los tiempos iniciales, si hay más de dos con el mejor resultado, se elige una secuencia de manera aleatoria.
2. Si las secuencias finales tienen la misma diferencia de los tiempos iniciales, se escoge la de menor tardanza, si hay más de dos con el mejor resultado, se elige una secuencia de manera aleatoria
3. En caso de que las secuencias tengan diferentes fitness, se utiliza la función de normalización para funciones multiobjetivo, los pesos w1 y w2 son determinados por quién maneja la programación.

- **Cálculo del fitness para diferencia de los tiempos iniciales y tardanza**

  - Se obtiene la nueva programación partiendo de la secuencia de programación y reemplazando desde la posición Bd hasta N el nuevo orden a analizar.
  - Se reordena los tiempos de procesamientos según la nueva secuencia.
  - Se identifica el trabajo donde ocurrió la interrupción y en la matriz de tiempos de liberación se modifica el nuevo tiempo de liberación para este trabajo. Se reordena esta matriz según la nueva secuencia.
  - Para calcular los tiempos de finalización de cada trabajo por etapa se utiliza otra función para calcular el CJ y funciona de la siguiente forma:
    - De la matriz de tiempos de finalización de cada trabajo por etapa de la programación inicial que se ingresó al inicio del algoritmo, toma los tiempos correspondientes desde el trabajo 1 hasta Bd-1
    - Teniendo en cuenta los tiempos de finalización de cada máquina por etapa luego del trabajo ubicado en Bd-1 de acuerdo con la información anterior, se procede a programar el resto de los trabajos con la nueva secuencia que va desde Bd hasta N. Para ello, se tiene en cuenta el nuevo tiempo de liberación del trabajo donde ocurrió la ruptura. En el caso de que se haya reprogramado anteriormente la secuencia por ruptura en otro(s) trabajo(s) se hace necesario incluir estos nuevos

tiempos de liberación en la matriz ingresada al inicio del algoritmo (los tiempos de liberación de cada trabajo en el paso 1).

- Como resultado se obtendrá una matriz con los nuevos tiempos de finalización por etapa de los trabajos reprogramados desde Bd hasta N. Con la cual se realiza lo siguiente:

– Teniendo en cuenta los tiempos de finalización de los trabajos en la primera etapa y los tiempos de procesamiento de cada trabajo por etapa se calcula los tiempos de inicio de los trabajos con la nueva programación.
– Luego se calcula el valor absoluto de las diferencias de los tiempos iniciales entre la programación inicial y la nueva programación. Se suman estos valores para obtener la diferencia total entre los tiempos de inicio.
– Al obtener los tiempos acumulados en cada etapa por trabajo, se obtiene el tiempo final de cada trabajo en la última etapa. Se obtiene el nuevo Makespan y además se calcula la tardanza de cada trabajo y luego se suman.
– Como resultado se obtiene un vector con la sumatoria de la diferencia de inicio de los tiempos iniciales, la tardanza total y el Makespan de la nueva secuencia de programación.

- **Tabu Search**

Este esquema es muy parecido al utilizado en el modelo de programación. Con esta función se evalúan por separado dos secuencias que parten de la secuencia inicial: 1) Es la misma secuencia inicial desde Bd hasta N y 2) Partiendo de la secuencia 1, se traslada el trabajo donde ocurre la interrupción de la programación hasta la posición final N.

En la secuencia anterior, se realizan varios intercambios entre dos trabajos hasta completar una matriz con 10 nuevas secuencias de los trabajos a reprogramar, de la misma forma que en el modelo de programación. La lista tabú comienza vacía hasta completar 3 movimientos prohibidos. Se escoge de estas la mejor secuencia calculando el fitness de todas y luego utilizando la fórmula de normalización se determina cuál de estas da los mejores resultados para minimizar

la sumatoria de la diferencia de los tiempos iniciales y la tardanza. Ejemplo para 20 trabajos con Bd=12:

Posición	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Reprog	14	20	15	6	3	1	7	2	12	8	10	19	16	13	9	4	18	5	17	11

Pos: 1 y 5	19	16	13	9	4	18	5	17	11
Nueva Sec	4	16	13	9	19	18	5	17	11

Figura 15. Representación de Tabu Search para el modelo de reprogramación.
Fuente: Elaboración propia.

Cada movimiento tabú permanece en la lista durante 3 iteraciones y las secuencias son evaluadas hasta completar las Wts iteraciones. Con esto se tiene una matriz con Wts secuencias, se evalúa el fitness y se utiliza la función de normalización para determinar cuál de estas da el mejor resultado para minimizar la sumatoria de la diferencia de los tiempos iniciales y la tardanza. La mejor secuencia encontrada serán los padres 1 o 2 para realizar el cruzamiento.

- **Cruzamiento padre 1 y padre 2**

  - Los padres 1 y 2 obtenidos de Tabu Search se someten a cruzamiento del mismo modo que en el modelo de programación.
  - Se toma una fracción del padre 1 que se inserta en el padre 2 de manera aleatoria de la misma forma que en el modelo de programación. Ejemplo de cruzamiento desde la posición Bd hasta N:

Posición	1	2	3	4	5	6	7	8	9
Padre 1	19	16	13	9	4	18	5	17	11
Padre 2	13	9	19	16	11	4	18	17	5

Posición	1	2	3	4	5	6	7	8	9
Hijo 1	*	*	19	16	11	18	5	17	*
Hijo 2	*	*	13	9	4	*	18	17	5

Posición	1	2	3	4	5	6	7	8	9
Hijo 1	13	9	19	16	11	18	5	17	4
Hijo 2	19	16	13	9	4	11	18	17	5

Figura 16. Representación cruzamiento de padre 1 y padre 2.

Fuente: Elaboración propia

– Se calcula el fitness a las dos secuencias hijos y con la fórmula de normalización se determina cuál de las dos secuencias tiene mejor resultado en términos de minimización de diferencia de los tiempos de inicio y tardanza.

- **Mutación**

– La ejecución de esta función está sujeta a una probabilidad determinada antes de ejecutar el algoritmo llamada ProbM. Se elige una fracción aleatoria de la secuencia hijo generada en el cruzamiento y se intercambian las posiciones. Ejemplo:

Posición	1	2	3	4	5	6	7	8	9
Hijo	13	9	19	16	11	18	5	17	4
Hijo M	13	9	19	16	11	18	17	4	5

Figura 17. Representación mutación secuencia hijo.

Fuente: Elaboración propia.

- Se escoge entre la secuencia hijo y la obtenida por la mutación la que presente mejores resultados en términos de minimización de la diferencia de los tiempos iniciales y la tardanza.

- **2 OPT**

Para finalizar, la secuencia anterior, ya sea el hijo inicial o la resultante de la mutación se somete a 2 OPT con el objetivo de intercambiar dos posiciones adyacentes y de esta forma buscar una mejor solución (Hasegawa, Ikeguchi, & Aihara, 1997), en caso de que la secuencia obtenida por 2 OPT sea mejor que la anterior en términos de minimización de las diferencias de tiempos iniciales y tardanza, se escoge esta última. Ejemplo:

Posición	1	2	3	4	5	6	7	8	9
Hijo	13	9	19	16	11	18	5	17	4
2 OPT	13	19	9	16	11	18	5	17	4

Figura 18. Representación 2 Opt.
Fuente: Elaboración propia.

8.6. Análisis del modelo de Reprogramación de la producción propuesto

En esta sección se realizará una comparación de modelos de programación de la producción encontrados en la literatura con los modelos de programación y reprogramación propuestos en esta investigación para medir el desempeño de estos:

8.6.1. Caso Mainieri y Ronconi (2013)

En esta investigación, se resuelve un problema de Flexible Flow Shop con una instancia pequeña de 6 trabajos, cuyo objetivo es minimizar las tardanzas. Para resolver este problema se utiliza un Algoritmo de programación de lista hacia atrás (Backward list scheduling algorithm) que se desarrolla en los siguientes pasos:

- **Paso 0.** Aplicar un método hacia adelante al problema original y obtener un programa inicial (Prog_Ini).
- **Paso 1.** Se construye el problema invertido donde se invierten las relaciones de precedencia entre las etapas y redefiniendo el tiempo de preparación y la fecha de vencimiento de cada trabajo, utilizando la información de Prog_Ini.
- **Paso 2.** Aplique el método hacia adelante al problema invertido utilizando el C_{ij} más grande de Prog_Ini como regla de prioridad.
- **Paso 3.** Invierta la secuencia de trabajos en cada máquina en la primera etapa.
- **Paso 4.** Para la primera etapa, busque un cronograma sin demoras desplazando los trabajos a la izquierda hasta que no haya tiempo de inactividad y sin violar las relaciones de precedencia y las secuencias obtenidas en el **Paso 3.**
- **Paso 5**. Para las etapas restantes, aplique el mismo método de avance del Paso 1.
- **Paso 6.** Se crea el cronograma final sin cambiar la relación de precedencia entre trabajos, se reasignan los trabajos a las máquinas en la primera etapa verificando si pueden comenzar antes. Se eliminan los posibles tiempos de inactividad generados en etapas posteriores.

A continuación, se presentan los datos del problema:

J	E1	E2	E3	Dd
1	2	5	4	14
2	1	5	3	11
3	3	4	1	10
4	1	3	5	11
5	2	3	1	7
6	4	5	5	18

Tabla 15. Tiempos (min) de proceso por etapa y tiempos de entrega de cada trabajo.
Fuente: Mainieri y Ronconi (2013)

Maquinas por etapa	E1	E2	E3
	2	2	1

Tabla 16. Máquinas por etapa.
Fuente: Mainieri y Ronconi (2013)

- **Resultados entre caso Mainieri y Ronconi (2013) y el algoritmo genético propuesto para programación de la producción de la presente investigación**

En la tabla 17 se pueden observar los resultados obtenidos del algoritmo genético propuesto comparados con la solución obtenida con el algoritmo de programación de lista hacia atrás. Para este caso se evalúo este problema con todas las opciones posibles para W1 y W2, se mantuvo fijo el número de iteraciones para Wag (algoritmo genético) y Wts (Tabu Search) y se mantuvo la población con n=10 para todos los casos.

De la tabla 17 se puede inferir que se obtuvo el mismo resultado de Makespan y tardanza que el caso de Mainieri y Ronconi con W1= 0.1 y W2= 0.9, donde se le dio el valor de peso más alto a la tardanza. Sin embargo, evaluando con las otras opciones de peso se encontró una secuencia con menor Makespan, pero un valor de tardanza más alto en solo una unidad. Con estos resultados podemos evidenciar la importancia de elegir los valores de W1 y W2 de acuerdo con las prioridades de la empresa como, por ejemplo, disminución de costos o satisfacción al cliente.

W1	W2	Wag	Wts	P	Makespan BW–EDD (minutos)	Tardanza BW-EDD (minutos)	Cmax Algoritmo Genetico propuesto (minutos)	Tardanza Algoritmo Genetico propuesto (minutos)	Diferencia en % Cmax	Diferencia en % tardanza	Tiempo	Secuencia algoritmo genetico propuesto
0.9	0.1	10	20	10	24	15	23	16	-4%	7%	15 Seg	4 5 3 2 1 6
0.8	0.2	10	20	10	24	15	23	16	-4%	7%	11 Seg	4 5 3 2 1 6
0.7	0.3	10	20	10	24	15	23	16	-4%	7%	10 Seg	4 5 3 2 1 6
0.6	0.4	10	20	10	24	15	23	16	-4%	7%	10 Seg	4 5 3 2 1 6
0.5	0.5	10	20	10	24	15	23	16	-4%	7%	18 Seg	4 5 3 2 1 6
0.4	0.6	10	20	10	24	15	23	16	-4%	7%	11 Seg	4 5 3 2 1 6
0.3	0.7	10	20	10	24	15	23	16	-4%	7%	17 Seg	4 5 3 2 1 6
0.2	0.8	10	20	10	24	15	23	16	-4%	7%	10 Seg	4 5 3 2 1 6
0.1	0.9	10	20	10	24	15	24	15	0%	0%	11 Seg	5 2 3 4 1 6

Tabla 17. Comparación de resultados entre caso Mainieri y Ronconi y algoritmo genético propuesto.
Fuente: Elaboración propia

- **Resultados con el modelo de reprogramación de la producción propuesto en la presente investigación**

De los resultados obtenidos con el modelo de programación (tabla 17), se escogió la secuencia 4 5 3 2 1 6 con Makespan de 23 y tardanza de 16 con una interrupción en la posición 3 (trabajo 3) y con un nuevo tiempo liberación de este trabajo en 8. Se obtuvieron los siguientes resultados:

Cmax Programación inicial: 23						Tardanza Programación inicial: 16						
W1	W2	Bd	Yk	Wag	Wts	Diferencia en los tiempos de inicio (minutos)	Tardanza (minutos)	Cmax (minutos)	Diferencia % tardanza	Diferencia % makespan	Tiempo	Secuencia
0.9	0.1	3	8	10	20	8	36	28	125%	22%	22 Seg	3 2 1 6
0.8	0.2	3	8	10	20	8	36	28	125%	22%	23 Seg	3 2 1 6
0.7	0.3	3	8	10	20	9	21	23	31%	0%	22 Seg	2 1 3 6
0.6	0.4	3	8	10	20	9	21	23	31%	0%	21 Seg	2 1 3 6
0.5	0.5	3	8	10	20	9	21	23	31%	0%	21 Seg	2 1 3 6
0.4	0.6	3	8	10	20	9	21	23	31%	0%	23 Seg	2 1 3 6
0.3	0.7	3	8	10	20	9	21	23	31%	0%	25 Seg	2 1 3 6
0.2	0.8	3	8	10	20	9	21	23	31%	0%	22 Seg	2 1 3 6
0.1	0.9	3	8	10	20	9	21	23	31%	0%	23 Seg	2 1 3 6

Tabla 18. Resultados con el modelo de reprogramación para el caso Mainieri y Ronconi.
Fuente: Elaboración propia

La secuencia escogida con el modelo de programación se evaluó con todas las opciones posibles de W1 y W2. Como se puede observar en la tabla 18 solo se obtuvieron 2 diferentes nuevas secuencias:

- 3-2-1-6 con Makespan de 28 y tardanza de 36. Esta nueva secuencia solo tuvo 8 (igual a yk) unidades de tiempo de diferencia de los tiempos de inicio con la secuencia de la programación inicial, una tardanza un 63% mayor que la obtenida con la secuencia inicial y con Makespan mayor con respecto a la secuencia inicial.
- 2-1-3-6 con Makespan de 23 y tardanza de 21. Esta nueva secuencia tuvo 9 unidades de tiempo de diferencia de los tiempos de inicio con la secuencia de la programación inicial, una tardanza un 31% mayor que la obtenida con la secuencia inicial y con Makespan igual que la secuencia inicial.

Para este caso la secuencia 2-1-3-6, representa la mejor opción y aunque debido a la interrupción haya aumentado la tardanza con respecto a la secuencia inicial, se podría negociar con los clientes la entrega de estos pedidos fuera del tiempo estipulado ofreciéndoles algún beneficio, como algún descuento adicional debido al retraso.

8.6.2. Caso Paternina, Montoya, Acero y Herrera (2008)

En esta investigación se resuelve un problema para Flexible Flow Shop para minimizar el Makespan basado en la teoría de restricciones. Como un ejemplo para exponer el método que proponen se busca darle solución a un problema de 7 trabajos y 5 etapas. El método de solución consiste en:

- **Paso 1.** Se calcula el flujo promedio de cada etapa.
- **Paso 2.** Se identifica la etapa cuello de botella y se calcula el Rj de cada trabajo en esta etapa.

- **Paso 3.** Se calculan los tiempos de cola de la etapa cuello de botella de cada trabajo con la diferencia entre la sumatoria del flujo medio por etapa y la sumatoria de los tiempos de proceso de las etapas ubicadas después del cuello de botella.
- **Paso 4.** Se programan los trabajos en la etapa anterior a la etapa cuello de botella. Seguidamente se programan todos los trabajos en el resto de las etapas anteriores al cuello de botella hasta llegar a la etapa 1.
- **Paso 5.** Se programan los trabajos en las etapas siguientes desde la etapa cuello de botella hasta la etapa final.
- **Paso 6.** Para ajustar la programación global, todos los trabajos se desplazan hacia la derecha para que el primer trabajo se inicie en el momento cero. Finalmente, se calcula el Makespan.

A continuación, se presentan los datos del problema:

J	E1	E2	E3	E4	E5	
1	7	5	10	7	5	
2	10	7	10	10	7	
3	12	8	12	7	8	
4	8	7	8	8	5	
5	7	7	8	7	7	
6	5	8	15	5	8	
7	9	3	10	9	3	
Suma tiempo	58	45	73	53	43	**Sumatoria flujo medio:**
# Maq x etapa	3	2	2	3	2	
Flujo medio	19.33333	22.5	36.5	17.66667	21.5	117.5

Tabla 19. Tiempos del proceso.

Fuente: Paternina, Montoya, Acero y Herrera (2008)

Máquinas x etapa	**E1**	**E2**	**E3**	**E4**	**E5**
	3	2	2	3	2

Tabla 20. Número de máquinas por etapa.
Fuente: Paternina, Montoya, Acero y Herrera (2008)

- **Resultados entre caso Paternina, Montoya, Acero y Herrera y el algoritmo genético propuesto para programación de la producción de la presente investigación**

La secuencia obtenida por el modelo de programación propuesto usando algoritmo genético generó una secuencia con un valor de Makespan menor que el encontrado con el método basado en teoría de restricciones. Aunque la diferencia entre ambos métodos solo es un 5%, se puede evidenciar que el modelo de programación propuesto en la presente investigación presenta mejores resultados.

Wag	Wts	P	Makespan TOC-based heuristic (minutos)	Makespan Algoritmo Genetico propuesto (minutos)	Diferencia en % del algoritmo propuesto	Tiempo	Secuencia algoritmo genetico propuesto
10	20	10	65	62	-5%	21 seg	6 7 5 3 2 4 1

Tabla 21. Resultados con el modelo de programación de la producción para el caso Paternina, Montoya, Acero y Herrera.
Fuente: Elaboración propia

En este caso la programación obtenida solo se evaluó en base a obtener un menor Makespan ya que el problema no tenía información correspondiente a tiempos de entrega para evaluar la tardanza. Sin embargo, los tiempos de finalización obtenidos en cada trabajo serán usados como tiempos de entrega para evaluar este problema con el modelo de reprogramación.

- **Resultados con el modelo de reprogramación de la producción propuesto en la presente investigación**

Con la secuencia obtenida con el modelo de programación, se considero una interrupción en el trabajo ubicado en la posición 4 (trabajo 3) con un nuevo tiempo de liberación en 15. Con esta información se obtuvo los siguientes resultados:

Cmax Programación inicial: 62											
W1	W2	Bd	Yk	Wag	Wts	Diferencia en los tiempos de inicio (minutos)	Tardanza (minutos)	Cmax (minutos)	Diferencia en % Cmax	Tiempo	Secuencia
0.1	0.9	4	15	10	20	18	8	63	2%	25 seg	4 2 3 1
0.2	0.8	4	15	10	20	18	8	63	2%	26 seg	4 2 3 1
0.3	0.7	4	15	10	20	18	8	63	2%	21 seg	4 2 3 1
0.4	0.6	4	15	10	20	18	8	63	2%	25 seg	4 2 3 1
0.5	0.5	4	15	10	20	18	8	63	2%	25 seg	4 2 3 1
0.6	0.4	4	15	10	20	10	15	69	11%	24 seg	3 2 4 1
0.7	0.3	4	15	10	20	10	15	69	11%	26 seg	3 2 4 1
0.8	0.2	4	15	10	20	10	15	69	11%	23 seg	3 2 4 1
0.9	0.1	4	15	10	20	10	15	69	11%	27 seg	3 2 4 1

Tabla 22. Resultados reprogramación de la producción para el caso Paternina, Montoya, Acero y Herrera.
Fuente: Elaboración propia.

Los tiempos de entrega de cada trabajo se obtuvieron con los tiempos de finalización de cada trabajo en la secuencia de programación inicial, con esto se pudo calcular la tardanza ocasionada debido a la interrupción ocurrida en el trabajo 3. De acuerdo a los resultados de la tabla 22 sólo se encontraron dos soluciones:

- 4-2-3-1 con una diferencia en los tiempos de inicio de 18, una tardanza de 8 y un Makespan de 63. La diferencia del Makespan con respecto a la programación inicial sólo es del 3%. Estos resultados se obtuvieron dandole un mayor peso a la tardanza (W2).
- 3-2-4-1 con una diferencia en los tiempos de inicio de 10, una tardanza de 15 y un Makespan de 69. La diferencia del Makespan con respecto a la programación inicial es del 11%. Estos resultados se obtuvieron dandole un mayor peso a la diferencia de tiempos iniciales (W1).

Cabe resaltar que el trabajo 3 en la secuencia de programación inicial tenía un tiempo de inicio igual a 5, por lo cual era de esperarse que como minimo la diferencia de tiempo inicial fuera de 10. En la secuencia de reprogramación 4-2-3-1 aunque la diferencia de tiempo inicial fue mayor que la secuencia 3-2-4-1, obtuvo un mejor resultado en terminos de tardanza y Makespan .

8.6.3. Caso Jimenez (2012)

En esta investigación se aborda el problema de Flow Shop Flexible para mimizar Makespan con base al de menor Makespan y al peor individuo usando como metodo de solucion la implementación del Algoritmo Genético de Chu-Beasley, el cual consiste en:

- **Paso 1.** Se procesa la base de datos del problema: tiempos de procesamiento, número de máquinas por etapa, semilla, tamaño de la población, número de iteraciones o generaciones, tasa de mutación y tasa de aceptación.
- **Paso 2.** Se genera una población inicial con dos alternativas, una utilizando el algoritmo NEH y la otra con una población aleatoria.
- **Paso 3.** Se calcula de la población al mejor individuo.
- **Paso 4.** Para el proceso de selección se hacen dos torneos para elegir a los padres.
- **Paso 5.** Se realiza recombinación usando el método de Recombinación Parcialmente Compatible (PMX Partially Matched Crossover).
- **Paso 6.** Se realiza mutación si se cumple un criterio determinado, realizando un trueque entre los genes del individuo.
- **Paso 7.** Se realiza el proceso de aspiración con dos criterios: 1) Si el individuo tiene mejor función objetivo que el de peor calidad, este lo reemplaza. 2) Se utiliza una tasa de aceptación para permanecer con el individuo de peor calidad.

A continuación, se presentan los datos del problema:

Se evaluaron 6 casos de esta investigación, donde se obtuvo la información correspondiente a los tiempos de procesamiento de cada trabajo por etapa y el número de máquinas por etapa. Se compararon los resultados de esta investigación con la propuesta en la presente investigación, aunque ambos están basados en el Algoritmo Genético de Chu-Beasley tienen diferencias en su método.

Con los resultados obtenidos del modelo de programación de la producción de la presente investigación se pudo calcular los tiempos de finalización de los trabajos de la secuencia de la programación inicial de cada caso, para posteriormente ser utilizados como tiempos de entrega en el modelo de reprogramación.

- **Resultados entre 6 casos evaluados por Jiménez y el algoritmo genético propuesto para programación de la producción de la presente investigación**

De los resultados obtenidos en la tabla 23 se puede observar que en todos los casos el algoritmo genetico propuesto en esta investigación presento resultados similares o muy cercanos a los resutados obtenidos por Jimenez (2012). A excepción del caso 2, ningún resultado de cada caso obtenido por el modelo de programación propuesto tiene una diferencia mayor a 0.4% en terminos de minimización del Makespan .

Cabe resaltar que en el caso 2 puede existir un error en la información de los tiempos de procesamiento por etapa o en el resultado final del Makespan del problema, debido a que el trabajo con mayor tiempo de procesamiento es igual a 310, por lo cual el valor del Makespan no puede ser inferior a este valor, y en el caso de Jimenez dio 304.

Casos	Cantidad de trabajos	Máquinas x etapa	Wag	Wts	P	Makespan Población inicial aleatoria (minutos)	Makespan Población inicial Heur NEH (minutos)	Makespan Algoritmo Genetico propuesto (minutos)	Diferencia en % del algoritmo propuesto	Tiempo
Caso 1	5	5 1	10	20	10	267	267	267	0.0%	10 seg
Caso 2	5	5 4 3 2 3	10	20	10	304	304	322	5.9%	14 seg
Caso 3	20	2 1 5 4 3	100	20	10	1170	1170	1170	0.0%	182 seg
Caso 4	20	1 5	100	20	10	1288	1288	1288	0.0%	47 seg
Caso 5	100	1 2	100	20	10	4911	4911	4929	0.4%	7.3 min
Caso 6	100	4 2 1 5 4	100	20	10	4985	4985	5000	0.3%	8.38 min

Tabla 23. Resultados con el modelo de programación de la producción para 6 casos evaluados por Jiménez (2012).
Fuente: Elaboración propia

- **Resultados entre 6 casos evaluados por Jiménez y el algoritmo genético propuesto para reprogramación de la producción de la presente investigación**

Con los resultados obtenidos con el modelo de programación se evaluó las secuencias de programación con una interrupción en una posición aleatoria dentro de estas secuencias. Los tiempos de finalización de cada trabajo obtenidos con el modelo de programación se utilizaron como los tiempos de entrega para el modelo de reprogramación y de esta forma poder calcular la tardanza.

Cada caso se evaluó en base a dos escenarios, por un lado, se le dio mayor peso a W2(tardanza) y, por otro lado, se le dio mayor peso a W1(diferencia de los tiempos de inicio). Para el caso 1 y 2 con una cantidad de trabajos igual a 5 se usó 10 iteraciones para Wag (Algoritmo genético) y 20 iteraciones para Wts (Tabu Search), y para los casos 3 en adelante con 20 trabajos o más se usó Wag con 100 iteraciones y Wts se mantuvo con 20. A continuación, se presentan los resultados obtenidos con el modelo de reprogramación:

Casos	W1	W2	Cmax Prog inicial	Bd	Yk	Wag	Wts	Diferencia tiempos de inicio (minutos)	Tardanza (minutos)	Cmax (minutos)	Diferencia en % de makespan	Tiempo
Caso 1	0.1	0.9	267	3	89	10	20	89	0	267	0.0%	22 seg
	0.9	0.1	267	3	89	10	20	89	0	267	0.0%	18 seg
Caso 2	0.1	0.9	322	2	80	10	20	80	106	322	0.0%	74 seg
	0.9	0.1	322	2	80	10	20	80	106	322	0.0%	83 seg
Caso 3	0.1	0.9	1170	6	850	100	20	1336	825	1179	0.8%	3.33 min
	0.9	0.1	1170	6	850	100	20	1295	663	1203	2.8%	3.71 min
Caso 4	0.1	0.9	1288	11	615	100	20	50	50	1293	0.4%	1.18 min
	0.9	0.1	1288	11	615	100	20	170	89	1293	0.4%	1.63 min
Caso 5	0.1	0.9	4929	52	2350	100	20	5603	2621	4929	0.0%	4.88 min
	0.9	0.1	4929	52	2350	100	20	3425	1341	4929	0.0%	4.71 min
Caso 6	0.1	0.9	5000	77	1230	100	20	818	633	5052	1.0%	7.65 min
	0.9	0.1	5000	77	1230	100	20	732	937	5000	0.0%	7.7 min

Tabla 24. Resultados con el modelo de reprogramación de la producción para 6 casos evaluados por Jiménez (2012).
Fuente: Elaboración propia

Para los **casos 1 y 2**, debido a que ambos en la primera etapa tienen 5 máquinas y 5 trabajos para programar, la diferencia de los tiempos de inicio fue igual al nuevo tiempo de liberación del trabajo donde ocurrió la interrupción. Para la tardanza, en el primer caso no hubo tardanza con respecto a la programación inicial pero el segundo caso si presento una tardanza de 106.

Cabe resaltar que, aunque en ambos casos se evaluaron los dos escenarios cambiando los pesos W1 y W2, esto no influyo en los resultados. Sin embargo, para ambos casos el Makespan se mantuvo igual que el de la programación inicial pese a la interrupción.

Para el **caso 3**, con una interrupción en el trabajo ubicado la posición 6 de la secuencia de programación, se puede inferir que ambos escenarios nos dan dos opciones de secuencias con resultados distintos para la función objetivo. Para el primer escenario donde se le da mayor peso a la tardanza, tanto la diferencia de los tiempos iniciales y la tardanza fueron mayores que en el escenario 2. Sin embargo, el primer escenario dio un resultado un poco mejor en términos de Makespan ya que solo es un 0.8% mayor que el Makespan obtenido de la secuencia de la programación inicial, mientras que el escenario 2 es mayor un 2.8% con respecto a éste.

Para el **caso 4**, con una interrupción en el trabajo 19 por no disponibilidad de materia prima según el orden de la programación, en el primer escenario la secuencia de reprogramación decide conservar el orden de la programación ya que solo genera una diferencia en los tiempos de inicio de 50 y una tardanza de 50. Por otro lado, para el segundo escenario, donde se le dio mayor peso a la diferencia de tiempos iniciales, su resultado fue mayor tanto para la diferencia de los tiempos iniciales como para la tardanza con respecto al primer escenario, esto es debido a que el algoritmo tiene componentes aleatorios. Sin embargo, en ambos casos el resultado del Makespan es el mismo con una diferencia de solo 0.4% de la programación inicial.

Para el **caso 5**, de acuerdo con los resultados se puede inferir que en el primer escenario pese a que se le dio un mayor peso a la tardanza, el resultado de éste fue mayor tanto para las diferencias de los tiempos iniciales y la tardanza con respecto a los resultados del segundo

escenario. Aunque ambos escenarios tienen un mismo resultado en términos de Makespan, el segundo escenario presenta la mejor solución para este caso.

Para el **caso 6**, como era de esperarse, el resultado de la tardanza en el escenario 1 fue mejor que el del escenario 2, pero en términos de diferencia de tiempos iniciales el segundo escenario presenta un mejor resultado que el primero. Con respecto al Makespan, el segundo escenario dio un resultado igual al de la programación inicial, por su parte, el segundo escenario solo tiene una diferencia de 1%.

CAPITULO III: CASO DE ESTUDIO EN UNA FÁBRICA DE CALZADO

En esta sección, se evaluará el modelo de reprogramación de la producción para un caso de una fábrica de calzado ubicada en la ciudad de Barranquilla en la ejecución de su plan de producción con un horizonte de una semana, la cual presenta interrupciones dentro de su proceso productivo debido a la no disponibilidad de materia prima. Además, se propone una modificación al modelo de reprogramación propuesto con el fin de obtener soluciones en un menor tiempo de ejecución del algoritmo.

9. Descripción de la problemática y condiciones de incertidumbre del sector.

Una de las problemáticas del sector de calzado en Colombia relacionadas con su productividad y competitividad es la falta de herramientas tecnológicas que le permitan el uso de técnicas de planificación, programación y control de la producción. Por lo cual, a los líderes dentro del proceso productivo se les dificulta realizar una gestión correcta de la producción que permita hacer un uso eficiente de sus recursos generando retrasos en la entrega de los trabajos, tiempos muertos, sobrecostos por uso de tiempo extra, entre otros (Ortiz Trian & Caicedo Rolón, 2014).

La consecución de materias primas es uno de los factores que más afectan la producción de calzado en Colombia, por lo tanto, dentro del horizonte de planeación debe considerarse la interrupción del programa de producción por falta de materias primas o como se propone en esta investigación, contar con un esquema de reprogramación de la producción con el fin de obtener una nueva programación que sea lo más parecida posible a la programación inicial con el fin de evitar cambios bruscos dentro del proceso productivo (Restrepo J., 2014).

A continuación, se presenta un caso de estudio de una fábrica de calzado ubicada en la ciudad de Barranquilla con una producción diaria de 2000 pares que presenta interrupción dentro de su sistema productivo debido a la no disponibilidad de materia prima ocasionada por retraso en el tiempo estipulado de llegada.

9.1. Presentación de la instancia.

Dentro del proceso productivo de esta fábrica de calzado una de las actividades que requieren mayor atención es la gestión de capelladas debido a que la moda es cambiante generando diversidad en los estilos y formas de las capelladas. Las capelladas en esta fábrica se realizan con materiales de tipo cuero sintético, gamuzas, corcho, entre otros; además, requiere para su producción hilos, hebillas, ojetes, adornos metálicos, etc.

El proceso de compras de esta fábrica no cuenta con un programa que le permita obtener información veraz de su inventario. La revisión de materia prima se realiza de manera manual y de acuerdo con esto se entrega la información al área de producción para organizar la programación de la producción de acuerdo a los pedidos y su meta diaria de producción. Sumado a esto, dentro del proceso de compras actual ocurre que algunos materiales de una misma referencia pueden llegar incompletos o en su defecto no llegan en la fecha prevista, por lo cual se hace necesario revisar la programación de la producción de esa semana y tomar decisiones para no afectar el flujo del proceso, su meta productiva, las entregas al cliente final, ni generar sobre costos por uso de tiempo extra.

Una referencia de un tipo de calzado se divide en varios lotes de acuerdo con el número total de pares pedidos (entre 50 a 300 pares) y en promedio cada lote tiene de a 18 a 21 pares. Cuando no llega a la fábrica alguna materia prima de alguna referencia, o llega incompleta, generalmente no se entrega esa referencia para el área de producción afectando así la programación estipulada. En ese caso los lotes de esta referencia no pueden ser procesados y deben ser cambiados por otra referencia distinta afectando el resto de las áreas de la fábrica ya que estas se programan de acuerdo al orden de procesamiento de las capelladas y los balanceos de línea del área de montaje.

Teniendo en cuenta que no es común que falte una materia prima que afecte toda la referencia completa, si no que puede afectar un solo lote de ésta, se propone conservar las referencias dentro del plan de producción semanal que está en ejecución y reprogramar solo el lote afectado con el modelo propuesto en esta investigación. Es necesario que la nueva programación se

obtenga antes de ingresar la referencia afectada al área de producción y que el tiempo de ejecución del algoritmo no sea superior a una hora, con el objetivo de trabajar en orden y no afectar el flujo del proceso de la etapa de corte (etapa inicial del proceso), ni de las etapas subsiguientes.

El proceso de capelladas incluye las siguientes operaciones o etapas:

- Corte: Operación Hombre-Máquina
- Control de calidad y conteo de piezas: Operación Manual
- Armado y guarnición: Operación Hombre-Máquina
- Control de calidad de capellada: Operación manual

Cada etapa cuenta con los siguientes recursos o máquinas:

Máquina o recurso por etapa	Etapa 1	Etapa 2	Etapa 3	Etapa 4
	3	1	5	3

Tabla 25. Número de recursos o máquinas por etapa.
Fuente: Elaboración propia.

Cada trabajo representa un lote de zapatos entre 18 - 21 pares y tiene un horizonte de planeación de una semana para un total de 10000 pares. Ver anexo 5

9.2. Análisis de resultados

9.2.1. Análisis de resultados con el modelo de programación de la producción propuesto

Se ejecutó el modelo de programación de dos formas, los resultados se pueden observar en la tabla 26:

- Con un número de iteraciones para algoritmo genético (W_{ag}) de 5000 y para Tabu Search (W_{ts}) de 20, y se evaluaron con todas las opciones para W_1 y W_2.
- Con un número de iteraciones para algoritmo genético (W_{ag}) de 1000 y para Tabu Search (W_{ts}) de 20, y se evaluaron con todas las opciones para W_1 y W_2.

Pesos		Iteración 5000			Iteración= 1000		
W1	W2	Cmax (minutos)	Tardanza (minutos)	Tiempo	Cmax (minutos)	Tardanza (minutos)	Tiempo
0.1	0.9	2796	48952	47.7222222	2802	49358	9.4 horas
0.2	0.8	2800	52060	28.8266667	2778	49891	9.8 horas
0.3	0.7	2788	50216	29.2125	2806	54138	9.2 horas
0.4	0.6	2789	51181	28.9	2789	54487	9.2 horas
0.5	0.5	2777	57094	28.8552778	2789	57020	9.2 horas
0.6	0.4	2780	59473	46.3 horas	2779	65492	9.3 horas
0.7	0.3	2777	63393	54.1 horas	2777	55953	9.3 horas
0.8	0.2	2775	65897	47.5 horas	2776	63126	9.5 horas
0.9	0.1	2773	75039	54.6 horas	2773	77158	9.3 horas

Tabla 26. Resultados con el modelo de programación de la producción para una fábrica de calzado.
Fuente: Elaboración Propia

De los resultados anteriores se puede inferir que los resultados para W_{ag}= 5000 no muestran una mejora significativa a comparación de los resultados dados por W_{ag}= 1000. Por lo tanto, para efectos prácticos dentro del sistema de producción se usará los resultados obtenidos con W_{ag}= 1000 ya que se puede obtener la programación de la producción en un día de trabajo.

Iteración= 1000						
Mejor Cmax: 2773				Mejor tardanza: 49358		
W1	W2	Cmax	Diferencia Cmax %	Tardanza	Diferencia tardanza %	Tiempo
0.1	0.9	2802	1.0%	49358	0%	9.4 horas
0.2	0.8	2778	0.2%	49891	1%	9.8 horas
0.3	0.7	2806	1.2%	54138	10%	9.2 horas
0.4	0.6	2789	0.6%	54487	10%	9.2 horas
0.5	0.5	2789	0.6%	57020	16%	9.2 horas
0.6	0.4	2779	0.2%	65492	33%	9.3 horas
0.7	0.3	2777	0.1%	55953	13%	9.3 horas
0.8	0.2	2776	0.1%	63126	28%	9.5 horas
0.9	0.1	2773	0.0%	77158	56%	9.3 horas

Tabla 27. Resultados con el modelo de programación de la producción para una fábrica de calzado con Wag= 1000.
Fuente: Elaboración propia

De los resultados de la tabla 27 se observa que para el caso de W_1 (Makespan)= 0.9 y W_2 (tardanza)= 0.1, el valor del Makespan fue el mejor para el resto de posibles opciones, y para el caso de W_1= 0.1 y W_2=0.9, el valor de la tardanza fue el mejor con respecto a las otras opciones. En base a estos mejores resultados se evaluó el resto de las opciones para elegir alguna que tuviera un buen resultado para Makespan y tardanza, la cual fue para W_1= 0.2 y W_2=0.8. La secuencia obtenida con ésta se elegirá la secuencia inicial de programación con un Makespan de 2778 y una tardanza de 49891. La secuencia es:

21- 255- 304- 119-16-45- 221-44- 137- 185- 152- 114-59- 327- 171- 311- 198-85- 239- 101-329- 382- 193- 265- 469- 146- 132- 184-51- 432- 173- 121- 134- 454-23-28-31- 443- 246- 115- 349- 278- 388-12-54- 313- 451-73- 444-46- 488- 106- 176- 100- 330-65- 320- 305-19- 210- 340- 296- 17-40- 303- 402- 335- 162-10- 144-36-99- 363- 383- 298- 178- 490-30- 288- 154-35- 220-70- 487- 300-18- 261- 170- 386-88- 33- 127- 413- 252- 190- 408-74- 227- 142- 341- 430- 422-55- 151- 323- 301- 371- 192-26- 485- 289-98- 479- 438-83- 310- 123- 360- 229- 159- 373- 374- 322- 6- 457- 4- 370- 219- 366- 228- 109- 280- 344-80- 407- 128- 392-69-50- 165- 314-29- 243- 436- 188- 437- 358- 460- 233- 196- 290- 315- 456- 181-27- 462- 308- 445-53- 412-60- 424- 471-91-92- 20-58- 293-94- 368-68- 321- 287- 143- 187- 105- 7- 133- 441-37- 11-66-25- 234-76- 274- 295- 183- 226- 279-64- 254- 419- 111- 476- 153- 120- 201- 470- 194- 248- 236- 214- 446- 244- 240- 428-57- 224- 139- 238-72-78- 384- 425- 205-49- 345- 271- 138- 464-41- 346- 404- 364- 47- 258- 281- 116- 450- 350- 202- 332- 131- 493-38- 461- 286- 307-63- 337- 197- 140- 395- 333- 199- 223- 150- 292- 276-

338-39- 448- 400- 372- 52- 222- 391-32- 317- 324- 186-13- 168- 177- 270-61-22- 2-14- 325- 285-482- 3- 117-93- 379- 145- 415- 376- 158- 1-95- 136-43- 297- 398- 182- 207- 427- 387- 273- 478-200- 299- 5- 312- 104- 426- 163- 195- 362- 253- 263- 365- 113- 495-71- 231- 172- 213- 347- 124-272- 467- 155- 357- 481- 135-79- 262- 107- 256-97- 125-24- 217- 179-48- 257- 75- 294- 122- 275-375-96- 334- 393-56- 475-87- 352- 203- 473- 318- 209- 268- 414- 241- 147- 459- 103- 316- 291-361- 453- 112- 249- 164- 242- 351- 431- 359- 355-67- 410- 189- 161- 235- 166- 406- 434-89- 466-141- 483- 405- 342- 416- 433- 191-62- 343- 156- 306- 237- 225- 282- 331- 411- 339- 245- 266-458- 353- 442- 212- 283- 498- 9- 277- 149- 206- 429- 284- 401- 260- 174- 397- 169- 326- 130-102- 118- 8- 449- 455- 250- 160- 385- 348- 380- 465- 496- 477- 269- 216- 356- 302- 409- 167-468- 175- 264- 390- 463-42- 230- 259- 148- 474- 367- 500- 394- 447- 108- 492- 497- 423- 378-319- 208- 110- 369-34- 421- 403- 377- 180-15- 484- 309- 218- 396- 381- 129- 417- 232-77- 389-126- 328- 354- 489- 499- 472- 491- 486- 440- 204- 251- 211- 399- 267- 452- 435- 215- 247- 336-480- 420-82- 157-86- 418- 84- 439-90- 494-81

9.2.2. Análisis de resultados con el modelo de reprogramación propuesto

Para evaluar el modelo de reprogramación para el caso de la fábrica de calzado, de manera aleatoria, se introdujeron 5 interrupciones en el horizonte de programación de una semana. Los parámetros utilizados para correr el modelo fueron los siguientes:

- Número de iteraciones para el algoritmo genético (W_{ag})= 100
- Número de iteraciones para Tabu Search (W_{ts})=20
- Peso para la diferencia de los tiempos iniciales (W_1)= 0.9
- Peso para la tardanza total (W_2)= 0.1

Cmax Programación inicial: 2778					**Tardanza Programación inicial: 49891**			
Wag=100			**Wts=20**		**W1=0.9**		**W2=0.1**	
Bd: Posición	**Bx: Trabajo**	**yk: Release time**	**Cmax (minutos)**	**Diferencia Cmax %**	**Diferencia Tiempo inicial (minutos)**	**Tardanza (minutos)**	**Diferencia Tardanza %**	**Tiempo**
85	300	612.78	2777	-0.04%	3252	49683	-0.42%	0,65 horas
137	69	1400.3	2781	0.11%	5385	49001	-1.78%	0,6 horas
251	448	2188.5	2783	0.18%	3223	48971	-1.84%	0,5 horas
311	347	2496.5	2779	0.04%	3437	49329	-1.13%	0,43 horas
476	440	2600.5	2781	0.11%	189	49423	-0.94%	0,39 horas

Tabla 28. Resultados con el modelo de reprogramación de la producción para el caso de una fábrica de calzado.
Fuente: Elaboración propia.

En la primera interrupción ocurrida en el trabajo ubicado en la posición 85 de la secuencia de la programación inicial, por ejemplo, se puede observar que, aunque la nueva secuencia de programación presenta diferencia entre los tiempos de inicio hubo una pequeña mejoría en las métricas de tardanza y Makespan con respecto a la programación inicial. Sin embargo, cabe resaltar que de los 416 trabajos que tuvieron que ser reprogramados, 215 conservaron su posición inicial, 194 trabajos se adelantaron una posición con respecto a la programación inicial y solo 7 trabajos tienen una posición totalmente diferente a la de la programación inicial. Al comparar la nueva secuencia de programación obtenida por la segunda interrupción con la secuencia obtenida luego de la primera interrupción, como otro ejemplo, se observó que, de los 364 trabajos a reprogramar, 144 conservaron la misma posición, 212 se adelantaron solo una posición y solo 8 trabajos tienen una nueva posición (2 de estos solo se adelantaron dos posiciones).

A partir de la segunda interrupción se puede observar que, aunque se le dio mayor peso a la diferencia de los tiempos iniciales, los resultados para la tardanza total tuvieron mejoría con respecto al resultado de la programación inicial. Por su parte los valores de Makespan se mantuvieron muy cercanos con relación al resultado obtenido en la programación inicial. Pese a que muchos trabajos no se entregaran en el tiempo estimado, esta fábrica no incurre en costos por penalización debido a que suple a sus 2 clientes internos. Sin embargo, el pronóstico de entrega generado a partir de este modelo es valioso para el área comercial con el fin de promover los productos.

Por la naturaleza de funcionamiento de esta fábrica es fundamental que no haya cambios bruscos dentro de la programación inicial ya que el resto de las áreas dentro de la empresa se programan de acuerdo con el flujo de la capellada ya que es posible que afecte el área de ensamble y por ende la entrega a sus clientes internos. Por lo tanto, la nueva secuencia presentada con este modelo de programación presenta una buena solución y una herramienta para continuar con el flujo del proceso dentro de la fábrica.

Con el desarrollo de este algoritmo de reprogramación se puede considerar otro tipo de incertidumbres diferentes al retraso de llegada de materias primas, como material incompleto o problemas de calidad del material. Además, es un aporte para futuras investigaciones sobre programación estocástica o programación robusta.

9.3. Modificación del modelo de reprogramación propuesto

Con el objetivo de obtener un buen resultado para este tipo de problemas en menos tiempo, se modificó el algoritmo del modelo de reprogramación de la siguiente forma:

- Se creó en Matlab la función para normalizar, en el modelo anterior no estaba presente como una función, si no como una formula dentro del cuerpo del algoritmo general.
- Los padres iniciales antes del cruzamiento se obtienen igual que el algoritmo anterior. Estos padres se evalúan en el algoritmo con un número de iteraciones denominado $\boldsymbol{W_{pi}}$ que determina la persona encargada del sistema de producción, con el objetivo de obtener la mejor secuencia de todas las W_{pi} generadas. Se evalúan con todas las funciones: Tabu Search, cruzamiento, mutación y 2-Opt. Esta secuencia es por el momento la mejor solución encontrada para este tipo de problema.
- Los padres iniciales nuevamente se evalúan en el algoritmo con todas sus funciones con un numero de iteraciones determinado por W_{ag}. Dentro de esta sección del algoritmo se incluye la función **break** de MATLAB con el objetivo de evaluar las secuencias obtenidas hasta WAG/X, es decir, para evaluar una parte de la secuencia antes del número total de iteraciones según W_{ag}. Si esta segunda secuencia obtenida con el uso de condiciones de finalización es mejor que la encontrada en el punto 1, el algoritmo se detiene. De lo contrario, el algoritmo continúa hasta evaluar el total de W_{ag} iteraciones, compara la mejor secuencia obtenida con todas las W_{ag} iteracciones con la secuencia obtenida en el punto 1 y selecciona la mejor entre estas dos.

A continuación, se presentan los resultados obtenidos con esta modificación del algoritmo:

- Número de iteraciones para solución inicial (W_{pi})= 50

- Número de iteraciones para el algoritmo genético (W_{ag})= 100
- Fracción de W_{ag} (X)= 2
- Número de iteraciones para Tabu Search (W_{ts})=20
- Peso para la diferencia de los tiempos iniciales (W_1)= 0.9
- Peso para la tardanza total (W_2)= 0.1

Para la ejecución de cada escenario, se partió de la secuencia inicial para la programación de la producción y las nuevas secuencias obtenidas luego de cada interrupción con el modelo anterior. Los resultados se presentan a continuación:

Cmax Programación inicial: 2778					**Tardanza Programación inicial: 49891**			
Bd: Posición	**Bx: Trabajo**	**yk: Release time**	**Cmax (minutos)**	**Diferencia Cmax %**	**Diferencia Tiempo inicial (minutos)**	**Tardanza (minutos)**	**Diferencia Tardanza %**	**Tiempo**
85	300	612.78	2777	-0.04%	5843	49910	0.04%	0,49 horas
137	69	1400.3	2778	0.00%	4466	49564	-0.66%	0,46 horas
251	448	2188.5	2782	0.14%	3335	49726	-0.33%	0,37 horas
311	347	2496.5	2783	0.18%	2020	49262	-1.26%	0,55 horas
476	440	2600.5	2782	0.14%	193	49290	-1.20%	0,2 horas

Tabla 29.Resultados con el modelo de reprogramación de la producción para el caso de una fábrica de calzado.
Fuente: Elaboración propia

Los resultados obtenidos con este modelo modificado no presentaron mejorías con respecto a las métricas de tardanza o de Makespan, siguen siendo muy similares a los resultados obtenidos por la programación inicial y al modelo anterior (tabla 28). Para el caso de la diferencia en los tiempos de inicio se puede evidenciar que para la primera interrupción en la posición 85 hay una diferencia entre ambos resultados de 1775 minutos, para la interrupción 2 a 4 hubo mejoría y para la última interrupción una diferencia de 42.

Diferencia Tiempo inicial reprogramación (minutos)	Diferencia Tiempo inicial reprogramación modificada (minutos)	Diferencia entre ambos modelos
4068	5843	1775
5035	4466	-569
4104	3335	-769
2602	2020	-582
151	193	42

Tabla 30. Diferencia entre los resultados de los modelos de reprogramación.
Fuente: Elaboración propia

En cuanto al tiempo de ejecución, este segundo modelo presento los resultados en un menor tiempo para casi todos los casos excepto en la cuarta interrupción, que tuvo un tiempo similar al del modelo anterior. Por lo tanto, se puede inferir que, en el caso de requerir una nueva programación con un menor tiempo de ejecución, este segundo modelo presenta buenos resultados para ejecutar un nuevo orden de producción teniendo en cuenta que esta fábrica suple a sus clientes internos y no cuenta con penalización por entrega de trabajos fuera del tiempo programado.

10. CONCLUSIONES

A continuación, se presentarán las conclusiones de acuerdo con los objetivos planteados en el desarrollo de esta investigación:

1. Se logró identificar y cuantificar los elementos con incertidumbre que pueden afectar un plan de producción, dentro de estos eventos inesperados se encuentran: Averías de máquinas, Llegada y/o cancelación de trabajos, Tiempos de proceso, Pedidos urgentes, Retraso en la llegada o escases de materiales y Reproceso o problemas de calidad

2. Se construyó el estado el arte sobre modelos de programación y reprogramación existentes en condiciones de incertidumbre donde se identificó:

- El tipo de ambiente de programación de cada investigación, dentro de los cuales se destacan con mayor presencia Job Shop con 29% y single machine con 19%, y para el caso de Flexible Flow Shop (FFS) solo contó con un porcentaje de 4%.
- Los diferentes tipos de incertidumbre de los modelos de programación y reprogramación entre los que se destacan averías de máquinas con un 38%, llegada y/o cancelación de pedidos con un 24%, por su parte para el caso de estudio de esta investigación el lead time solo corresponde a un 3%.
- Para cada ambiente de programación se identificó el método de solución utilizado. Los métodos de solución utilizados tanto para modelos de reprogramación con incertidumbre en lead time como los que son de tipo FFS incluyen reglas de despacho, heurísticas y metaheurísticas.

3. De acuerdo con la revisión de literatura de esta investigación se puede inferir que existe una brecha en las investigaciones en modelos de reprogramación de la producción cuando hay perturbación debido a la incertidumbre en el lead time en la adquisición de materias primas. Además, en lo que respecta a los ambientes de programación, se evidencio que existe una brecha en modelos de reprogramación de la producción en problemas de asignación de máquina utilizando Flexible Flow Shop. Se puede destacar que ninguna de las investigaciones de reprogramación de la producción en ambientes tipo Flexible Flow Shop considera el lead time como el tipo de incertidumbre que afecta su plan de producción y en lo que se refiere a su función objetivo, aunque incluyen minimización de Makespan o tardanza, ninguno considera una función bi-objetivo que permita la minimización de estas dos métricas.

4. La presente investigación permitió desarrollar un modelo de reprogramación bi-objetivo (minimización de Makespan y Tardanza) de la producción como respuesta a una

interrupción en el proceso de aprovisionamiento de materias primas. Para obtener este modelo, utilizando Tabu Search y el algoritmo genético de Chu-Beasley , se desarrolló en MATLAB un modelo inicial de programación para Flow shop para minimización de Makespan, y en base a éste, se desarrolló el modelo de programación de la producción bi-objetivo para minimizar Makespan y tardanza total que incluye una función de normalización que permite escoger la mejor solución para este problema, luego, en base al modelo anterior e incluyendo una nueva función de 2-Opt, se logró desarrollar el modelo de reprogramación de la producción bi-objetivo que minimiza la sumatoria de la diferencia de los tiempos de inicio de los trabajos entre la programación inicial y la nueva secuencia, y a su vez, minimiza la tardanza total.

5. Se evidenció que el modelo de programación de la producción al ser comparado con las instancias de Taillard y otros métodos utilizados en este tipo de problema encontrados dentro de la literatura, con problemas desde 5 a 100 trabajos para minimizar el Makespan o tardanza, obtuvo resultados similares y, en algunos casos, mejores que estos. Partiendo de los resultados del modelo de programación se simuló el uso del modelo de reprogramación con interrupción por retraso en el tiempo de llegada de materia prima para un trabajo, obteniendo como resultado una nueva programación que aunque presenta diferencias entre los tiempos de inicio con respecto a la programación inicial, como era de esperarse, y aumento en la tardanza, en algunos casos, los resultados de Makespan de la nueva programación son similares y en algunos casos iguales al Makespan de la programación inicial.

Se aplicó el modelo de programación y el de reprogramación de la producción para un caso de estudio de una fábrica de calzado para la actividad de elaboración de capelladas. En los resultados del modelo de reprogramación hubo diferencia entre los tiempos de inicio de los trabajos con respecto a la programación inicial, sin embargo, cabe resaltar que las diferencias entre la tardanza y el Makespan con respecto a la programación inicial fueron mínimas. Aunque la fábrica suple a sus clientes internos y no cuenta con costo de

penalización por entrega fuera del tiempo estimado, la nueva de la secuencia de la programación de la producción obtenida es una herramienta para programar el resto de las áreas de la fábrica y poder brindar un mejor pronóstico de entrega al cliente final.

De acuerdo con la evidencia de los resultados, se puede concluir que el presente modelo permite mejorar la confiabilidad sobre las métricas de Makespan y de tardanza de un programa de producción pese a que ocurran interrupciones en el proceso de aprovisionamiento de materias primas.

6. Por último, con respecto a las investigaciones futuras, este modelo puede ser de gran utilidad no solo a interrupciones ocasionadas por el retraso de materias primas, puesto que, se puede considerar interrupciones por material incompleto o problemas de calidad del material. Por otro lado, podría considerarse la minimización en la diferencia de los tiempos de finalización de los trabajos entre la programación inicial y la nueva secuencia generada y que a su vez minimice la tardanza, con el fin de obtener la menor cantidad de trabajos retrasados.

Además, esta investigación puede ser un aporte significativo para investigaciones referentes a programación robusta y programación estocástica. Para el caso de programación robusta, teniendo en consideración que el cronograma debería ser insensible ante interrupciones que ocurran dentro del sistema, se puede considerar, por ejemplo, que, dada una interrupción, los trabajos sin concluir sean desplazados solo lo necesario con respecto al tiempo de inicio de la programación inicial para adaptarse a la interrupción, y a su vez, respetando los tiempos de liberación de cada trabajo. Para el caso de programación estocástica, se podría considerar, obtener la información correspondiente a la confiabilidad de los proveedores en cuanto al cumplimiento de entregas de materias primas y obtener su distribución de probabilidad, con lo cual se podría realizar un modelo que incluya incertidumbre por incumplimiento por parte del

proveedor generando interrupción en el programa de producción por la no disponibilidad de materia prima, material incompleto o por problemas de calidad.

11. REFERENCIAS

Abumaizar, R., & Svestka, J. (1997). Rescheduling job shops under random disruptions. *International Journal of Production Research, 35*(7), 2065-2082.

Acevedo Chedid, J., Salas Navarro, K., Ospina Mateus, H., & Santander Mercado, A. (2017). Reprogramación de producción en cadenas de suministro colaborativas: Una revisión de la literatura. *Espacios*, 30.

Adhitya, A., Srinivasan, R., & Karimi, I. (2004). A heuristic reactive scheduling strategy for recovering from refinery supply chain disruptions. *Laboratory for Intelligent Applications in Chemical Engineering Department of Chemical and Biomolecular Engineering, National University of Singapore*.

Akturk, M., & Gorgulu, E. (1999). Match-up scheduling under a machine breakdown. *European journal of operational research*, 81-97.

Alagöz, O., & Azizoğlu, M. (2003). Rescheduling of identical parallel machines under machine eligibility constraints. *European Journal of Operational Research*, 523-532.

Al-Hinai, N., & ElMekkawy, T. (2011). Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. *International Journal of Production Economics*, 279-291.

Arnaout, J. P. (2010). Rescheduling of Parallel Machines with Setup Times using Simulation. *In IIE Annual Conference. Proceedings* (p. 1). Norcross, United States: Institute of Industrial and Systems Engineers (IISE).

Arnaout, J. P. (2014). Rescheduling of parallel machines with stochastic processing and setup times. *Journal of Manufacturing Systems, 33(3),* , 376-384.

Arnaout, J., & Rabadi, G. (2008). Rescheduling of unrelated parallel machines under machine breakdowns. *International Journal of Applied Management Science*, 75-89.

Aytug, H., Lawley, M., McKay, K., Mohan, S., & Uzsoy, R. (2005). Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research., 161*(1), 86-110.

Azizoglu, M., & Alagöz, O. (2005). Parallel-machine rescheduling with machine disruptions. *IIE Transactions, 37(12)*, 1113-1118.

Babtiste, P., & Favrel, J. (1993). Taking into account the rescheduling problem during the scheduling phase. *Production planning & Control*, 349-360.

Bajestani, M., & Beck, J. (2013). Scheduling a Dynamic Aircraft Repair Shop. *Journal of Artificial Intelligence Research, 47*, 35-70.

Barbarosoglu, G., & Ozgur, D. (1999). A tabu search algorithm for the vehicle routing problem. *Computers & Operations Research, 26*(3), 255-270.

Baykasoğlu, A., & Karaslan, F. (2017). Solving comprehensive dynamic job shop scheduling problem by using a GRASP based approach. *International Journal of Production Research, 55(11),*, 3308-3325.

Bean, J., Birge, J., Mittenthal, J., & Noon, C. (1991). Matchup scheduling with multiple resources, release dates and disruptions. *Operations Research*, 470-483.

Ben-Daya, M., & Al-Fawzan, M. (1998). A tabu search approach for the flow shop scheduling problem. *European journal of operational research, 109*(1), 88-95.

Bidot, J., Laborie, P., Beck, J., & Vidal, T. (2003). Using simuation for execution monitoring and on-line rescheduling with uncertain durations. *Plan Execution (ICAPS workshop) The 13th International Conference on Automated Planning & Scheduling*, (pp. 16-23). Trento, Italy.

Bierwirth, C., & Mattfeld, D. (1999). Production scheduling and rescheduling with genetic algorithms. *Evolutionary computation*, 1-17.

Boiteux, O. D., Corominas, A., & Lus, A. (2007). Estado del arte sobre planificación agregada de la producción. *Universitat Politècnica de Catalunya*.

Bougeret, M., Alves Pessoa, A., & Poss, M. (2018). Robust scheduling with budgeted uncertainty. *Elsevier B.V.*

Brucker, P. (2006). *Scheduling Algorithm* (5 ed.). Germany: Springer.

Buddala, R., & Mahapatra, S. (2019). Two-stage teaching-learning-based optimization method for flexible job-shop scheduling under machine breakdown. *The International Journal of Advanced Manufacturing Technology*, 1419-1432.

Byeon, E., Wu, S., & Storer, R. (1998). Decomposition heuristics for robust job-shop scheduling. *IEEE Transactions on Robotics and Automation*, 303-313.

Chang, J., & Luh, Y. (1997). Integration of scheduling and control in a job shop. *Journal of the Chinese Institute of Engineers*, 67-76.

Chang, S., & Hsieh, F. (1992). Order and production scheduling rescheduling for flow shops. *Proceedings 1992 IEEE International Conference on Robotics and Automation* (pp. 1173-1178). Nice, France, France: IEEE.

Chapman, S. (2006). *Planificación y control de la producción.* Mexico: Pearson educación.

Choi, S., & Wang, K. (2012). Flexible flow shop scheduling with stochastic processing times: A decomposition-based approach. *Computers & Industrial Engineering*, 362-373.

Christy, D., & Kanet, J. (1988). Open order rescheduling in job shops with demand uncertainty: a simulation study. *Decision Sciences*, 801-818.

Chu, P., & Beasley, J. (1997). A genetic algorithm for the generalised assignment problem. *Computers & Operations Research, 24*(1), 17-23.

Church, L. K., & Uzsoy, R. (1992). Analysis of periodic event-driven rescheduling policies in dynamic shops. *International Journal of Computer Integrated Manufacturing*, 153-163.

Conway, R., Miller, L., & Maxwell, W. (2003). *Theory of Scheduling.* Dover.

Corominas, A., & Pastor, R. (2010). Replanning working time under annualised working hours. *International Journal of Production Research*, 1493-1515.

Cowling, P., & Johansson, M. (2002). Using real time information for effective dynamic scheduling. *European journal of operational research*, 230-244.

Dai, M., Tang, D., Giret, A., Salido, M., & Li, W. (2013). Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. *Robotics and Computer-Integrated Manufacturing, 29*(5), 418-429.

Daniels, R., & Carrillo, J. (1997). β-Robust scheduling for single-machine systems with uncertain processing times. *IIE transactions*, 977-985.

Daniels, R., & Kouvelis, P. (1995). Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science*, 363-376.

DINERO. (2017, Septiembre 22). *Revista Dinero*. Retrieved from https://www.dinero.com/economia/articulo/costos-y-tiempo-que-tarda--importar-y-exportar-una-mercancia-en-colombia/250158

DINERO. (2018, 4 12). *Revista Dinero*. Retrieved from https://www.dinero.com/edicion-impresa/negocios/articulo/regulacion-aduanera-y-de-zonas-francas-genera-incertidumbre/257200

Dolgui, A., & Ould-Louly, M. (2002). A model for supply planning under lead time uncertainty. *International Journal of Production Economics*, 145-152.

Dong, Y.-H., & Jang, J. (2012). Production rescheduling for machine breakdown at a job shop. *International Journal of Production Research*, 2681-2691.

Dupon., A., Van Nieuwenhuyse, I., & Vandaele, N. (2002). The impact of sequence changes on product lead time. *Robotics and Computer-Integrated Manufacturing*, 327-333.

Fang, J., & Xi, Y. (1997). A rolling horizon job shop rescheduling strategy in the dynamic environment. *The International Journal of Advanced Manufacturing Technology*, 227-232.

Farn, C., & Muhlemann, A. (1979). The dynamic aspects of a production scheduling problem. *International Journal of Production Research*, 15 - 21.

Gahm, C., Kanet, J., & Tuma, A. (2019). On the flexibility of a decision theory-based heuristic for single machine scheduling. *Computers & Operations Research*, 103-115.

Gao, K., Suganthan, P., Pan, Q., Tasgetiren,, M., & Sadollah, A. (2016). Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion. *Knowledge-based systems*, 1-16.

Gao, K., Yang, F., Zhou, M., Pan, Q., & Suganthan, P. (2018). Flexible Job-Shop Rescheduling for New Job Insertion by Using Discrete Jaya Algorithm. *IEEE transactions on cybernetics*, 1944-1955.

García, M. C., Marquez, G. P., & Burtseva, L. (2015). Rescheduling in Industrial Environments: Emerging Technologies and Forthcoming Trends. *International Journal of Combinatorial Optimization Problems and Informatics, 6*(3), 34-48.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of Flowshop and Fobshop scheduling. *Mathematics of operations research*, 117-129.

Gomes, M., Barbosa-Póvoa, A., & Novais, A. (2010). A discrete time reactive scheduling model for new order insertion in job shop, make-to-order industries. *International Journal of Production Research*, 7395-7422.

Guo, B., & Nonaka, Y. (1999). Rescheduling and optimization of schedules considering machine failures. *International Journal of Production Economics*, 503-513.

Gürel, S., & Cincioğlu, D. (2015). Rescheduling with controllable processing times for number of disrupted jobs and manufacturing cost objectives. *International Journal of Production Research, 53(9),*, 2751-2770.

H Leung, S. C., Wu, Y., & Lei, K. K. (2003). Multi-site aggregate production planning with multiple objectives: a goal programming approach. *Taylor and Francis*, 425–436.

Hall, N., & Potts, C. (2010). Rescheduling for job unavailability. *Operations research, 58*(3), 746-755.

Hall, N., & Potts, C. (2010). Rescheduling for job unavailability. *Operations research, 58*(3), 746-755.

Hall, N., Liu, Z., & Potts, C. (2007). Rescheduling for multiple new orders. *INFORMS Journal on Computing*, 633-645.

Hasegawa, M., Ikeguchi, T., & Aihara, K. (1997). Combination of chaotic neurodynamics with the 2-opt algorithm to solve traveling salesman problems. *Physical Review Letters, 79*(12), 2344.

Honghong, Y., & Zhiming, W. (2003). The application of adaptive genetic algorithms in FMS dynamic rescheduling. *International Journal of Computer Integrated Manufacturing*, 382-397.

INVIAS. (2019, JULIO 16). *INSTITUTO NACIONAL DE VIAS*. Retrieved from https://www.invias.gov.co/index.php/component/content/article/2-uncategorised/57-estado-de-la-red-vial

Jaegler, Y., Jaegler, A., Burlat, P., & Lamouri, S. (2018). The ConWip production control system: a systematic review and classification. *International Journal of Production Research*, 5736-5757.

Jain, A., & Elmaraghy, H. (1997). Production scheduling/rescheduling in flexible manufacturing. *International Journal of Production Research*, 281-309.

Jiang, Y., Zhou, X., & Chen, Y. (2019). A practical production-distribution rescheduling model with conflict and unexpected disruptions. *IEEE Access*, 38523-38534.

Kadlec, P., Gabrys, B., & Strandt, S. (2009). Data-driven soft sensors in the process industry. *Computers & chemical engineering*, 795-814.

Katragjini, K., Vallada, E., & Ruiz, R. (2013). Flow shop rescheduling under different types of disruption. *International Journal of Production Research, 51(3),*, 780-797.

Keshavarz, T., & Salmasi, N. (2013). Makespan minimisation in flexible flowshop sequence-dependent group scheduling problem. *International Journal of Production Research, 51*(20), 6182-6193.

Khiong Yang, K., & Jacobs, F. R. (1999). Replaning the Master Production Schedule for a Capacity- Constrained Job Shop. *Decision Sciencies*, 719.

Kim, M., & Kim, Y. (1994). Simulation-based real-time scheduling in a flexible manufacturing system. *Journal of manufacturing Systems*, 85-93.

Kumar, P. (1994). Scheduling manufacturing systems of re-entrant line. *In Stochastic modeling and analysis of manufacturing systems*, 325-360.

Lanza, G., Stricker, N., & Moser, R. (2013). Concept of an intelligent production control for global manufacturing in dynamic environments based on rescheduling. *In 2013 IEEE International Conference on Industrial Engineering and Engineering Management* (pp. 315-319). Bangkok, Thailand: IEEE.

Larsen, R., & Pranzo, M. (2019). A framework for dynamic rescheduling problems. *International Journal of Production Research*, 16-33.

Lazo Eche, E., Gutierrez Segura, F., & Vergara Moreno, E. (2016). Un algoritmo eficiente para problemas single machine con tiempos de procesamiento difusos. *Revista Cubana de Ciencias Informáticas*, 139-153.

Lee, C., Leung, J., & Yu, G. (2006). Two machine scheduling under disruptions with transportation considerations. *Journal of Scheduling*, 35-48.

Lee, C., Piramuthu, S., & Tsai, Y. (1997). Job shop scheduling with a genetic algorithm and machine learning. *International Journal of production research, 35*(4), 1171-1191.

Lee, T., & Loong, Y. (2019). A review of scheduling problem and resolution methods in flexible flow shop. *International Journal of Industrial Engineering Computations, 10*(1), 67-88.

Lendínez, L. C. (2019). Kanban. Metodología para aumentar la eficiencia de los procesos. *3C Tecnologia*, 30-41.

Leon, V., J., Wu, S., D., & Storer, R. (1994). Robustness measures and robust scheduling for job shops. *IIE transactions*, 32-43.

Leung, S., & Wu, Y. (2004). A robust optimization model for stochastic aggregate production planning. *Production planning & control, 15*(5), 502-514.

Li, H., Li, Z., Li, L., & Hu, B. (2000). A production rescheduling expert simulation system. *European Journal of Operational Research*, 283-293.

Li, J., Pan, Q., & Mao, K. (2015). A discrete teaching-learning-based optimisation algorithm for realistic flowshop rescheduling problems. *Engineering Applications of Artificial Intelligence*, 279-292.

Li, R., Shyu, Y., & Adiga, S. (1993). A heuristic rescheduling algorithm for computer-based production scheduling systems. *The International Journal Of Production Research*, 1815-1826.

Li, X., Peng, Z., Du, B., Guo, J., Xu, W., & Zhuan, K. (2017). Hybrid artificial bee colony algorithm with a rescheduling strategy for solving flexible job shop scheduling problems. *Computers & Industrial Engineering*, 10-26.

Liu, L., Gu, H., & Xi, Y. (2007). Robust and stable scheduling of a single machine with random machine breakdowns. *The International Journal of Advanced Manufacturing Technology*, 645-654.

Liu, X., Shen, Z., & Dai, L. (2018). Scheduling optimization of flexible flow shop. . *In 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC)* (pp. 1652-1657). IEEE.

Liu, Z., & Ro, Y. (2014). Rescheduling for machine disruption to minimize makespan and maximum lateness. *Journal of Scheduling, 17(4),*, 339-352.

Luh, P. B., & Feng, W. (2003). *From Manufacturing Scheduling to Supply Chain Coordination: The Control of Complexity and Uncertainty.* IEEE, Montreal. Retrieved Junio 18, 2018, from https://ieeexplore.ieee.org/document/1595144/

Lv, S., & Qiao, L. (2014). Process planning and scheduling integration with optimal rescheduling strategies. *International Journal of Computer Integrated Manufacturing*, 638-655.

Ma, Z., Yang, Z., Liu, S., & Wu, S. (2018). Optimized rescheduling of multiple production lines for flowshop production of reinforced precast concrete components. *Automation in Construction*, 86-97.

Malakooti, B. (2011). Systematic decision process for intelligent decision making. *Journal of Intelligent Manufacturing, 22*(4), 627-642.

McKay, K., & Wiers, V. (2003). Planning, scheduling and dispatching tasks in production control. *Cognition, Technology & Work*, 82-93.

Mehta, S., & Uzsoy, R. (1998). Predictable scheduling of a job shop subject to breakdowns. *IEEE Transactions on Robotics and Automation*, 365-378.

Morales, A. P. (2012). *Solución del problema de programación de flow-shop Flexible empleando el algoritmo genético de chu-beasley.* Pereira: Universidad Tecnológica de Pereira.

Muhlemann, A. P., Lockett, A. G., & Farn, C. K. (1982). Job shop scheduling heuristics and frequency of scheduling. *The International Journal of Production Research, 20*(2), 227-241.

N. Mustafizul, K., & S. Mainul, H. (2002). Possible Cost Reduction By Applyng MRP in a Transformer Manufacturing Company of Bangladesh. *Proceedings of the Int. Conf. on Manufacturing*, (pp. 259-267). Dhaka. Retrieved Junio 25, 2018, from http://irep.iium.edu.my/26862/1/MRP_%2308.pdf

Naseri, E., & Kuzgunkaya, O. (2010). Cost-based rescheduling in a flexible manufacturing system using filtered-beam search. *In IIE Annual Conference. Proceedings* (p. 1). Institute of Industrial and Systems Engineers (IISE).

Nawaz, M., Enscore Jr, E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega, 11*(1), 91-95.

Nouiri, M., Bekrar, A., Jemai, A., & Ammari, A. (2018). A new rescheduling heuristic for flexible job shop problem with machine disruption. *Service Orientation in Holonic and Multi-Agent Manufacturing*, 461-476.

Ocampo, E., Grisales, Y., & Echeverri, M. (2006). Algoritmo genético modificado aplicado al problema de secuenciamiento de tareas en sistemas de producción lineal-Flow Shop. *Scientia et Technica, 12*(30), 285-290.

O'Donovan , R., Uzsoy, R., & McKay, K. (1999). Predictable scheduling of a single machine with breakdowns and sensitive jobs. *International Journal of Production Research*, 4217-4233.

Ortiz Trian, V. K., & Caicedo Rolón, A. J. (2014). Programación óptima de la producción en una pequeña empresa de calzado-en Colombia. *Ingeniería Industrial, 35*(2), 114-130.

Paternina, C. D., Montoya, J. R., Acero, M. J., & Herrera, M. C. (2008). Scheduling jobs on a k-stage flexible flow-shop. *Annals of Operations Research, 164*(1), 29-40.

Peng, K., Pan, Q., Gao, L., Li , X., Das, S., & Zhang, B. (2019). A multi-start variable neighbourhood descent algorithm for hybrid flowshop rescheduling. *Swarm and Evolutionary Computation, 45*, 92-112.

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research*, 3202-3212.

Pfeiffer, A., Kádár, B., Csáji, B., & Monostori, L. (2005). Simulation supported analysis of a dynamic rescheduling system. *Manufacturing, Modelling, Management and Control*, 25-30.

Potts, C., & Strusevich, V. (2009). Fifty years of scheduling: a survey of milestones. *Journal of the Operational Research Society*, S41 - S68.

Rahmani, D., & Ramezanian, R. (2016). A stable reactive approach in dynamic flexible flow shop scheduling with unexpected disruptions: A case study. *Computers & Industrial Engineering*, 360-372.

Rangsaritratsamee, R., Ferrell Jr, W., & Kurz, M. (2004). Dynamic rescheduling that simultaneously considers efficiency and stability. *Computers & Industrial Engineering, 46(1),*, 1-15.

Restrepo J., S. B. (2014). Sistema de gestión de producción para la empresa SCARPA CALZADO ORIGINAL en la ciudad de Bogotá, Colombia.

Rodammer, F., & White, K. (1988). A recent survey of production scheduling. *IEEE transactions on systems, man, and cybernetics*, 841-851.

Sabuncuoglu , I., & Bayız, M. (2000). Analysis of reactive scheduling problems in a job shop environment. *European Journal of operational research*, 567-586.

Sabuncuoglu, I., & Kizilisik, O. (2003). Reactive scheduling in a dynamic and stochastic FMS environment. *International Journal of Production Research*, 4211-4231.

Salazar Hornig, E., & Medina S, J. (2013). Minimizatión del makespan en máquinas paralelas idénticas con tiempos de preparación dependientes de la secuencia utilizando un algoritmo genético. *Ingeniería, investigación y tecnología*, 43-51.

Salido, M., Escamilla, J., Barber, F., & Giret, A. (2016). Rescheduling in job-shop problems for sustainable manufacturing systems. *Journal of Cleaner Production*.

Sawik, T. (2007). Integer programming approach to reactive scheduling in make-to-order manufacturing. *Mathematical and Computer Modelling*, 1373-1387.

Sharma, S., & Singla, V. (2019). The Effects of Implementation of Kanban System on Productivity: A Case Study of Auto Parts Company. *IUP Journal of Operations Management*.

Shen, X., & Yao, X. (2015). Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems. *Information Sciences*, 198-224.

Shirai, K., & Amano, Y. (2014). Mathematical Analysis of Manufacturing Lead Time: Applications to Deadline Scheduling in Manufacturing Systems. *Electronics and Communications in Japan*, 24-34.

Silva, C., Sousa, J., & Runkler, T. (2008). Rescheduling and optimization of logistic processes using GA and ACO. *Engineering Applications of Artificial Intelligence*, 343-352.

Spearman, M., Woodruff, D., & Hopp, W. (1990). CONWIP: a pull alternative to kanban. *The International Journal of Production Research*, 879-894.

Sun, J., & Xue, D. (2001). A dynamic reactive scheduling mechanism for responding to changes of production orders and manufacturing resources. *Computers in Industry, 46*(2), 189-207.

Taillard, E. (1990). Some efficient heuristic methods for the flow shop sequencing problem. *European journal of Operational research*, 65-74.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European journal of operational research, 64*(2), 278-285.

THE WORLD BANK. (2018). *THE WORLD BANK*. Retrieved from https://lpi.worldbank.org/international

Unal, A., Uzsoy, R., & Kiran, A. (1997). Rescheduling on a single machine with part-type dependent setup times and deadlines. *Annals of Operations Research*, 93-113.

Valledor, P., Gomez, A., Priore, P., & Puente, J. (2018). Solving multi-objective rescheduling problems in dynamic permutation flow shop environments with disruptions. *International Journal of Production Research*, 6363-6377.

Vidal, C. J., & Goetschalckx, M. (2000). Modeling the effect of uncertainties on global logistics systems. *Journal of Business Logistics;*, 95.

Vieira, G. E., Herrmann, J. W., & Lin, E. (2003). Rescheduling Manufacturing Systems: A framework of strategies, policies and methods. *Journal of Scheduling*, 39-62.

Vieira, G., Herrmann, J., & Lin, E. (2000). Analytical models to predict the performance of a single-machine system under periodic and event-driven rescheduling strategies. *International journal of production research*, 1899-1915.

Vieira, G., Herrmann, J., & Lin, E. (2000). Predicting the performance of rescheduling strategies for parallel machine systems. *Journal of manufacturing Systems*, 256-266.

Wang, C., Jiang, P., & Lu, T. (2018). Production events graphical deduction model enabled real-time production control system for smart job shop. *Journal of Mechanical Engineering Science*, 2803-2820.

Wang, D., Yin, Y., & Cheng, , T. (2018). Parallel-machine rescheduling with job unavailability and rejection. *Omega*, 246-260.

Wong , T., Leung, C., Mak, K., & Fung, R. (2006). Integrated process planning and scheduling/rescheduling—an agent-based approach. *International Journal of Production Research*, 3627-3655.

Wong, T. (2005). An iterative genetic algorithm-based approach to machine assignment. *HKU Theses Online*.

World Economic Forum. (2017). *The Global Competitiveness Report 2017–2018.* Geneva: World Economic Forum.

Wu, H., & Li, R. (1995). A new rescheduling method for computer based scheduling systems. *International journal of production research*, 2097-2110.

Wu, S., Storer, R., & Pei-Chann, C. (1993). One-machine rescheduling heuristics with efficiency and stability as criteria. *Computers & Operations Research*, 1-14.

Xiao, Y., Yuan, Y., Konak, A., & Zhang, R. (2015). Non-permutation flow shop scheduling with order acceptance and weighted tardiness. *Applied Mathematics and Computation*, 312-333.

Yamamoto, M. &. (1985). Scheduling/rescheduling in the manufacturing operating system environment. *International Journal of Production Research*, 705-722.

Yang, B. (2007). Single machine rescheduling with new jobs arrivals and processing time compression. *The International Journal of Advanced Manufacturing Technology, *, 378-384.

Yu, S., & Pan, Q. (2012). A Rescheduling Method for Operation Time Delay Disturbance in Steelmaking and Continuous Casting Production Process. *Journal of Iron and Steel Research International*, 33-41.

Yuan, J., & Mu, Y. (2007). Rescheduling with release dates to minimize makespan under a limit on the maximum sequence disruption. *European Journal of Operational Research*, 936-944.

Zhang, J., & Qiao, C. (2015). A bi-objective model for robust resource-constrained project scheduling problem with random activity durations. *2015 IEEE 12th International Conference on Networking, Sensing and Control* (pp. 28 - 32). Taipei, Taiwan: IEEE.

Zhang, L., Gao, L., & Li, X. (2013). A hybrid genetic algorithm and tabu search for a multi-objective dynamic job shop scheduling problem. *International Journal of Production Research*, 3516-3531.

Zhang, S., & Wong, T. (2017). Flexible job-shop scheduling/rescheduling in dynamic environment: a hybrid MAS/ACO approach. *International Journal of Production Research*, 3173-3196.

Zhenyu, G., Linfeng, L., Jiajia, Z., & Guorong, L. (2016). A flexible job-shop rescheduling method by considering the machine equipment availability. *2016 Chinese Control and Decision Conference (CCDC)* (pp. 4898-4902). Yinchuan, China: IEEE.

Zhou, R., Nee, A., & Lee, H. (2009). Performance of an ant colony optimisation algorithm in dynamic job shop scheduling problems. *International Journal of Production Research, 47(11),*, 2903-2920.

Zuo, X., Mo, H., & Wu, J. (2009). A robust scheduling method based on a multi-objective immune algorithm. *Information Sciences*, 3359-3369.

Zweben, M., Davis, E., Daun, B., & Deale, M. (1993). Scheduling and Rescheduling with Iterative Repair. *IEEE Transactions on Systems, Man, and Cybernetics, 23(6),*, 1588-1596.

ANEXOS

1. **Modelo de programación de la producción para el problema de asignación de máquina de tipo Flow Shop**

Cálculo del fitness – Makespan

```
function [CmaxQ]= KQfitness (PT,P,N,M)
%Con esta función se calcula el Makespan  para cada secuencia
X=PT(P,:);
% X=[]
CJ(1,1)= X(1,1);%Tiempo que tarda en ejecutarse el trabajo 1 en la máquina 1
 for i= 2:N;
CJ(i,1)= CJ(i-1,1)+X(i,1);
 end
 for j= 2:M;
    CJ(1,j)= CJ(1,j-1)+X(1,j);
 end
 %Se calcula el CJ para cada máquina:
 for i=2:N;
    j=2;
    CJ(i,j)= max(CJ(i-1,j),CJ(i,j-1))+X(i,j);
 for i=2:N;
    j=3;
    CJ(i,j)= max(CJ(i-1,j),CJ(i,j-1))+X(i,j);
 end
 for i=2:N;
    j=4;
    CJ(i,j)= max(CJ(i-1,j),CJ(i,j-1))+X(i,j);
 end
  for i=2:N;
    j=5;
    CJ(i,j)= max(CJ(i-1,j),CJ(i,j-1))+X(i,j);
  end
   CmaxQ=CJ(N,M); %Makespan  de la secuencia

 end

 end
```

Tabu Search

```
function Padre= Tabusearch(secini,N,M,PT)
% Busqueda tabu para encontrar los Padres # 1 y 2
n=20;
A=100000;
for i=1:n;
   j=1:N;
   Padreini1= [secini];
   t=randperm(N,2);
   B1=find(t(1,1)==Padreini1);
   B2=find(t(1,2)==Padreini1);
   Padreini1(1,B1)=t(1,2);
   Padreini1(1,B2)=t(1,1);
   Padre1(i,j)=Padreini1(:,j);
 for j= 1:2;
      Perm(i,j)= t(:,j);
 end
end
for i= 1:n;
   j=1;
 CmaxP1(i,j)=[KQfitness(PT,Padre1(i,:),N,M)];
end
minP1=min(CmaxP1);
k= find(CmaxP1==minP1);
P1= Padre1(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padre1=P1(L,:);
LT= [k(L,1)];
Ltb1=Perm(LT,:);
Ltb2=[Ltb1(1,2), Ltb1(1,1)];
Listatabu1=[Ltb1; Ltb2];
for i=1:n;
   j=1:N;
   Padre1=P1(L,:);
   t=randperm(N,2);
    if t(1,1)==Ltb1(1,1)&t(1,2)==Ltb1(1,2);
       t(1,:)=[0];
    elseif t(1,1)==Ltb2(1,1)&t(1,2)==Ltb2(1,2);
t(1,:)=[0];
    end
     B1=find(t(1,1)==Padre1);
   B2=find(t(1,2)==Padre1);
   Padre1(1,B1)=t(1,2);
```

```
   Padre1(1,B2)=t(1,1);
    Pad2(i,j)=Padre1(:,j);

 for j= 1:2;
      Perm2(i,j)= t(:,j);
 end

end
for i= 1:n;
   j=1;
 CmaxP2(i,j)= [KQfitness(PT,Pad2(i,:),N,M)];
end
search=find(Perm2(:,1)==0);
if search>0;
   CmaxP2(search)=A;
end
minP2=min(CmaxP2);
k= find(CmaxP2==minP2);
P2= Pad2(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padre2=P2(L,:);
LT= [k(L,1)];
Ltb3=Perm2(LT,:);
Ltb4=[Ltb3(1,2), Ltb3(1,1)];
Listatabu2=[Listatabu1; Ltb3; Ltb4];
for i=1:n;
   j=1:N;
   Padre2=P2(L,:);
   t=randperm(N,2);
    if t(1,1)==Ltb1(1,1)&t(1,2)==Ltb1(1,2);
       t(1,:)=[0];
    elseif t(1,1)==Ltb2(1,1)&t(1,2)==Ltb2(1,2);
t(1,:)=[0];
 elseif t(1,1)==Ltb3(1,1)&t(1,2)==Ltb3(1,2);
t(1,:)=[0];
elseif t(1,1)==Ltb4(1,1)&t(1,2)==Ltb4(1,2);
t(1,:)=[0];
    end
     B1=find(t(1,1)==Padre2);
   B2=find(t(1,2)==Padre2);
   Padre2(1,B1)=t(1,2);
   Padre2(1,B2)=t(1,1);
```

```
    Pad3(i,j)=Padre2(:,j);

 for j= 1:2;
       Perm3(i,j)= t(:,j);
 end

end
for i= 1:n;
   j=1;
 CmaxP3(i,j)= [KQfitness(PT,Pad3(i,:),N,M)];
end
search=find(Perm3(:,1)==0);
if search>0;
   CmaxP3(search)=A;
end
minP3=min(CmaxP3);
k= find(CmaxP3==minP3);
P3= Pad3(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padre3=P3(L,:);
LT= [k(L,1)];
Ltb5=Perm3(LT,:);
Ltb6=[Ltb5(1,2), Ltb5(1,1)];
Listatabu3=[Listatabu2; Ltb5; Ltb6];

%Luego de obtener la lista tabu inicial, se pueden realizar n interacciones
%y se actualiza la lista tabu y el padre en cada corrida
for w=1:50;
   Padrepp=Padre3;
   Ltbf=Listatabu3;
for i=1:n;
   j=1:N;
   t=randperm(N,2);
    if t(1,1)==Ltbf(1,1)&t(1,2)==Ltbf(1,2);
       t(1,:)=[0];
    elseif t(1,1)==Ltbf(2,1)&t(1,2)==Ltbf(2,2);
t(1,:)=[0];
 elseif t(1,1)==Ltbf(3,1)&t(1,2)==Ltbf(3,2);
t(1,:)=[0];
elseif t(1,1)==Ltbf(4,1)&t(1,2)==Ltbf(4,2);
t(1,:)=[0];
elseif t(1,1)==Ltbf(5,1)&t(1,2)==Ltbf(5,2);
```

```
t(1,:)=[0];
elseif t(1,1)==Ltbf(6,1)&t(1,2)==Ltbf(6,2);
t(1,:)=[0];
    end
     B1=find(t(1,1)==Padrepp);
   B2=find(t(1,2)==Padrepp);
   Padrepp(1,B1)=t(1,2);
   Padrepp(1,B2)=t(1,1);
    Pad4(i,j)=Padrepp(:,j);

 for j= 1:2;
      Perm4(i,j)= t(:,j);
 end
end
for i= 1:n;
   j=1;
 CmaxP4(i,j)= [KQfitness(PT,Pad4(i,:),N,M)];
end
search=find(Perm4(:,1)==0);
if search>0;
   CmaxP4(search)=A;
end
minP4=min(CmaxP4);
k= find(CmaxP4==minP4);
P4= Pad4(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padrep=P4(L,:);
Padre3=Padrep;
LT= [k(L,1)];
Ltb7=Perm4(LT,:);
Ltb8=[Ltb7(1,2), Ltb7(1,1)];
Listatabu3(1,:)=[Ltbf(3,1),Ltbf(3,2)];
Listatabu3(2,:)=[Ltbf(4,1),Ltbf(4,2)];
Listatabu3(3,:)=[Ltbf(5,1),Ltbf(5,2)];
Listatabu3(4,:)=[Ltbf(6,1),Ltbf(6,2)];
Listatabu3(5,:)=[Ltb7(1,1), Ltb7(1,2)];
Listatabu3(6,:)=[Ltb8(1,1), Ltb8(1,2)];
j=1:20;
Padres(w,j)=Padrep(:,j);
j=1;
CmaxT(w,j)=minP4(:,j);
end;
```

```
minPadre=min(CmaxT);
k= find(CmaxT==minPadre);
P= Padres(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padre=Padres(L,:);
end
```

Cruzamiento de Padre #1 y Padre #2

```
function Desc= Crossover(parent1,parent2)
%Función para realizar el cruzamiento de dos padres para dos bloques
%en las posiciones 5 a 8 y en las posiciones 13 a 16. Esta función
%da como resultados dos descendientes.

cross1=(5:8);
 C11=parent1(cross1);
 C12=parent2(cross1);
parent1(5:8)=C12;
parent2(5:8)=C11;

cross2=(13:16);
 C21=parent1(cross2);
 C22=parent2(cross2);
parent1(13:16)=C22;
parent2(13:16)=C21;

CrossP1=[C11, C21];
CrossP2=[C12, C22];

parent1(5:8)=[200 200 200 200];
parent1(13:16)=[200 200 200 200];
for i=1:8;
X= find(CrossP2(1,i)==CrossP1);
if X>0;
   parent1=parent1;
   else
   B= find(CrossP2(1,i)==parent1);
   parent1(1,B)=100;
end
end
for i=1:8;
Y= find(CrossP1(1,i)==CrossP2);
```

```
if Y>0;
   parent1=parent1;
   else
   B= find(100==parent1);
   t=B(1,1);
   parent1(1,t)= CrossP1(1,i);
end
end
parent1(5:8)=C12;
parent1(13:16)=C22;
Desc1=parent1;
%Ahora se busca el Desc2 de la misma forma que se hizo con el Desc1


CrossP1=[C11, C21];
CrossP2=[C12, C22];

parent2(5:8)=[200 200 200 200];
parent2(13:16)=[200 200 200 200];
for i=1:8;
X= find(CrossP1(1,i)==CrossP2);
if X>0;
   parent2=parent2;
   else
   B2= find(CrossP1(1,i)==parent2);
   parent2(1,B2)=100;
end
end
for i=1:8;
Y= find(CrossP2(1,i)==CrossP1);
if Y>0;
   parent2=parent2;
   else
   B2= find(100==parent2);
   t2=B2(1,1);
   parent2(1,t2)= CrossP2(1,i);
end
end
parent2(5:8)=C11;
parent2(13:16)=C21;
Desc2=parent2;
Desc=[Desc1; Desc2];
end
```

Mutación

```
function HijoM= Mutacion(Hijo,N)
%Con esta función se somete a mutación el hijo obtenido del cruzamiento,
%intercambiando dos posciones dentro de la secuencia hijo.
 Sechijo= [Hijo];
    tm=randperm(N,2);
    Bm1=find(tm(1,1)==Sechijo);
    Bm2=find(tm(1,2)==Sechijo);
   Sechijo(1,Bm1)=tm(1,2);
   Sechijo(1,Bm2)=tm(1,1);
   HijoM=Sechijo;
 end
```

Algoritmo genético

```
clc, clear all
%La población inicial se realiza de manera aleatoria
%para elegir los padres
TP= 'Prueba1.xlsx';
PT= xlsread(TP);%Tiempo de procesamiento de cada trabajo
%en cada máquina
S= size(PT); %Tamaño de la matriz
N= S(1,1)%trabajos
M= S(1,2)%máquinas
p= 500; %tamaño de la población inicial
Pob1= Generar_pobl(N,p);
for i= 1:p;
    j=1;
 Cmax1(i,j)= [KQfitness(PT,Pob1(i,:),N,M)];
end
%De la población actual se escoge la secuencia con el menor Cmax
%(si hay más de 2 secuencias, se escoge una de manera aleatoria y se
% aplica Tabu Search para buscar el Padre inicial #1

min1=min(Cmax1);
k= find(Cmax1==min1);
Px1= Pob1(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padini1=Px1(L,:);
%Generamos P2 para obtener el Padre #2 con Tabu Search
Pob2= Generar_pobl(N,p);
```

```
for i= 1:p;
   j=1;
 Cmax2(i,j)= [KQfitness(PT,Pob2(i,:),N,M)];
end

%De la población actual se escoge la secuencia con el menor Cmax
%(si hay más de 2 secuencias, se escoge una de manera aleatoria y se
% aplica Tabu Search para buscar el Padre inicial #2

min2=min(Cmax2);
k= find(Cmax2==min2);
Px2= Pob2(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padini2=Px2(L,:);
for i=1:5000;
%Utilizamos búsqueda tabú para buscar el Padre #1 con
%secuencia inicial=Padini1, de igual forma para el Padre #2
%usando como secuencia inicial=Padini2;
Parent1= Tabusearch(Padini1,N,M,PT);
Parent2= Tabusearch(Padini2,N,M,PT);
Cmaxparent1= [KQfitness(PT,Parent1(1,:),N,M)];
Cmaxparent2= [KQfitness(PT,Parent2(1,:),N,M)];
%Realizamos cruce de los padres insertando 2 bloques de 4 trabajos del padre
%2 al padre 1 y viceversa, de esta forma obtenemos dos descendiente
Hijos= Cross2(Parent1, Parent2);
%Se evalúa el fitness de los dos hijos y se escoge el de menor Cmax
CmaxHijo1= KQfitness(PT,Hijos(1,:),N,M);
CmaxHijo2= KQfitness(PT,Hijos(2,:),N,M);
if CmaxHijo1>CmaxHijo2;
   Hijo=Hijos(2,:);
   CmaxHijo=CmaxHijo2;
else Hijo=Hijos(1,:);
   CmaxHijo=CmaxHijo1;
end

% Se realiza mutación con una probabilidad determinada
%Si al calcular el Makespan  de la secuencia de la mutación es mejor
%que el resultado del hijo, se conserva como solución la secuencia
%de la mutación, de lo contrario, permanece como solución la
%secuencia hijo.
pm= rand(1);
 if pm<0.4;
```

```
HijoM= Mutacion(Hijo,N);
 CmaxMut= KQfitness(PT,HijoM,N,M);
 if CmaxHijo>CmaxMut;
    Hijo=HijoM;
    CmaxHijo= CmaxMut;
 else Hijo=Hijo;
    CmaxHijo= CmaxHijo;
 end
 end
 j=1:20;
Besthijo(i,j)=Hijo(:,j);
j=1;
Cmaxbesthijo(i,j)=CmaxHijo(:,j)
end
%Se calcula el Cmax de todos los hijos y se escoge la secuencia
%con menor Cmax o Makespan
SolutCmaxBesthijo= min(Cmaxbesthijo)
kbest= find(Cmaxbesthijo==SolutCmaxBesthijo);
Best_hijo= Besthijo(kbest,:);
Kbest= size(kbest);
lbest= Kbest(1,1);
Lbest= randi(lbest);
Solbesthijo=Best_hijo(Lbest,:)
```

2. Modelo de programación de la producción para Flexible Flow Shop

Cálculo del fitness – Makespan y Tardanza total

```
function [CmaxFFSTmax]= KQFFSfitness (PT,P,N,ME,Dd)
% Con esta función se calcula el tiempo de finalización de cada trabajo
% para una secuencia determinada
 [A B]=size(ME);
Sum=0;
X=PT(P,:);
Dds= Dd(P,:);
%%%%%
 MEIn=ME(1,1);
   R(1,:)=[1 MEIn];
   for k=2:B;
      j=1:2;
      MEI=MEIn+1;
      MEF= MEI+ME(1,k)-1;
      MEIn=MEF;
```

```
      RR=[MEI MEF];
      R(k,j)=RR(:,j);
    end
     for i=1:B;
      Sum=Sum+ME(1,i);
     end
    for i=1:N;
      j=1:Sum;
      E1(i,j)=0;
    end
    %%%%Para trabajo 1 según la secuencia
    Xi=0;
    for i=1:B;
      R1=R(i,1);
      E1(1,R1)=Xi+X(1,i);
      Xi=E1(1,R1);
    end
    %%%% Para trabajos 2 hasta N en etapa 1
    for i=2:N;
      j=1:ME(1,1);
    E2= E1(i-1,j);
    MinE2= min(E2);
    FE1= find(MinE2==E2);
    SFE1=size(FE1);
    K=SFE1(1,2);
    l=randi(K);
    F=FE1(1,l);
  E1(i,j)= E1(i-1,j);
  E1(i,F)= E1(i-1,F)+X(i,1);
    end
  for i=2:N;
    E2=E1(i,:);
    for j=2:B;
    E2X= E1(i-1,R(j,1):R(j,2));
    MinE2X= min(E2X);
   FER= find(MinE2X==E2X);
   SFER=size(FER);
   K=SFER(1,2);
   l=randi(K);
  F=FER(1,l);
  EXB=E1(i,R(j-1,1):R(j-1,2));
  EXA=E1(i-1,R(j-1,1):R(j-1,2));
  YX=EXB==EXA;
  FYX=find(0==YX);
```

```
 CjA=EXB(1,FYX);
 CjB=E2X(1,F);
 if CjA>CjB;
    E2X(1,F)=CjA+X(i,j);
 else
    E2X(1,F)=CjB+X(i,j);
 end
E1(i,R(j,1):R(j,2))=E2X;
   end
 end
EP=R(B,1):R(B,2);
s=size(EP);
sf=s(1,2);
 for i=1:N;
  CjE=E1(i,EP);
  j=1:sf;
  CjEF(i,j)=CjE(:,j);
 end
 CJi(1)=max(CjEF(1,:));
  [s1 s2]=size(CjEF);
  if s2==1;
  CJiF=CjEF;
  else
  for i=2:N;
     BCj=CjEF(i-1,:)==CjEF(i,:);
     FBCj=find(0==BCj);
     CJi(i)=CjEF(i,FBCj);
  end
    CJiF=transpose(CJi);
  end
  Cjifx= CJiF;
for i=1:N;
   Tardxj= CJiF-Dds;
   if Tardxj(i,1)>0;
      Tardanza(i,1)=Tardxj(i,1);
   else
      Tardanza(i,1)=0;
   end
   TardT= transpose(Tardanza);
end
MaxCmaxt=max(CJiF);
CmaxFFSTmax= [MaxCmaxt, sum(TardT)];
% Tmax= sum(TardT)
 End
```

Generar población

```
%Generar población, se generan p secuencias de n!
function P= Generar_pobl(N,p)

for i= 1:p;
   T= [randperm(N)];
   for j=1:N;
P(i,j)= T(:,j);
   end
%    P= [Y]
end
end
```

Tabu Search

```
function PadreTS= TsearchFFS(secini,N,ME,PT,Dd,Wts,W1,W2)
% % % Busqueda tabu para encontrar los Padres # 1 y 2
n=10;
A=100000;
for i=1:n;
   j=1:N;
   Padreini1= [secini];
   t=randperm(N,2);
   B1=find(t(1,1)==Padreini1);
   B2=find(t(1,2)==Padreini1);
   Padreini1(1,B1)=t(1,2);
   Padreini1(1,B2)=t(1,1);
   Padre1(i,j)=Padreini1(:,j);
 for j= 1:2;
      Perm(i,j)= t(:,j);
 end
end
for i= 1:n;
 [CmaxTmaxP1]= [KQFFSfitness(PT,Padre1(i,:),N,ME,Dd)];
 for j=1:2;
MtCmaxTmaxP1(i,j)= CmaxTmaxP1(:,j);
 end
end
MaxCT1= max(MtCmaxTmaxP1);
MinCT1= min(MtCmaxTmaxP1);
if MaxCT1(1,2)==MinCT1(1,2);
```

```
   j=1;
   Cmaxx=MtCmaxTmaxP1(:,j);
   MinCmaxx=min(Cmaxx);
   k= find(MinCmaxx==Cmaxx);
Px1= Padre1(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Pad1=Px1(L,:);
end
   if MaxCT1(1,1)==MinCT1(1,1);
 j=2;
   Tmaxx=MtCmaxTmaxP1(:,j);
   MinTmaxx=min(Tmaxx);
   k= find(MinTmaxx==Tmaxx);
Px1= Padre1(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Pad1=Px1(L,:);
   end
if MaxCT1(1,2)~=MinCT1(1,2);
   if MaxCT1(1,1)~=MinCT1(1,1);
for i=1:n;
   j=1;
% Normalizamos funciones de Makespan  y Tardanza:
fCmax1(i,j)=(MaxCT1(1,1)- MtCmaxTmaxP1(i,1))/(MaxCT1(1,1)- MinCT1(1,1));
fTmax1(i,j)=(MaxCT1(1,2)- MtCmaxTmaxP1(i,2))/(MaxCT1(1,2)- MinCT1(1,2));
%Con la ecucación siguiente se determina cual de las secuencias da un
%mejor resultado. Se colocó un peso w del 0.4 para Makespan  y 0.6 para
%tardanza. La solución cuyo resultado sea mayor acercandose a uno, es la
%elegida como mejor secuencia.
U(i,j)= W1*fCmax1(i,1)+ W2*fTmax1(i,1);
end
MaxU=max(U);
k= find(MaxU==U);
Px1= Padre1(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Pad1=Px1(L,:);
   end
end
LT= [k(L,1)];
```

```
Ltb1=Perm(LT,:);
Ltb2=[Ltb1(1,2), Ltb1(1,1)];
Listatabu1=[Ltb1; Ltb2];
%%%%2
for i=1:n;
   j=1:N;
   Padre1=Px1(L,:);
   t=randperm(N,2);
    if t(1,1)==Ltb1(1,1)&t(1,2)==Ltb1(1,2);
       t(1,:)=[0];
    elseif t(1,1)==Ltb2(1,1)&t(1,2)==Ltb2(1,2);
t(1,:)=[0];
    end
     B1=find(t(1,1)==Padre1);
   B2=find(t(1,2)==Padre1);
   Padre1(1,B1)=t(1,2);
   Padre1(1,B2)=t(1,1);
    Pad2(i,j)=Padre1(:,j);

 for j= 1:2;
      Perm2(i,j)= t(:,j);
 end

end
for i= 1:n;
 [CmaxTmaxP2]= [KQFFSfitness(PT,Pad2(i,:),N,ME,Dd)];
 for j=1:2;
MtCmaxTmaxP2(i,j)= CmaxTmaxP2(:,j);
 end
end
 for i=1:n;
    j=1;
 Makespan max2(i,j)= MtCmaxTmaxP2(i,1);
 Makespan min2(i,j)= MtCmaxTmaxP2(i,1);
 Tardanzamax2(i,j)= MtCmaxTmaxP2(i,2);
 Tardanzamin2(i,j)= MtCmaxTmaxP2(i,2);
end
search=find(Perm2(:,1)==0);
if search>0;
   Makespan max2(search)=0;
   MaxMak2= max(Makespan max2);
   Tardanzamax2(search)=0;
   MaxTard2= max(Tardanzamax2);
   Makespan min2(search)=A;
```

```
   MinMak2=min(Makespan min2);
   Tardanzamin2(search)=A;
   MinTard2= min(Tardanzamin2);
else
  MaxMak2= max(Makespan max2);
  MinMak2= min(Makespan max2);
  MaxTard2= max(Tardanzamax2);
  MinTard2= min(Tardanzamax2);
end
if MaxTard2==MinTard2;
   j=1;
   Cmaxx2=MtCmaxTmaxP2(:,j);
   if search>0;
      Cmaxx2(search)=A;
   end
   MinCmaxx2=min(Cmaxx2);
   k= find(MinCmaxx2==Cmaxx2);
Px2= Pad2(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padre2=Px2(L,:);
end
 if MaxMak2==MinMak2;
   j=2;
   Tmaxx2=MtCmaxTmaxP2(:,j);
   if search>0;
      Tmaxx2(search)=A;
   end
   MinTmaxx2=min(Tmaxx2);
   k= find(MinTmaxx2==Tmaxx2);
Px2= Pad2(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padre2=Px2(L,:);
 end
if MaxTard2~=MinTard2;
   if MaxMak2~=MinMak2;
for i=1:n;
   j=1;
% Normalizamos funciones de Makespan  y Tardanza:
fCmax2(i,j)=(MaxMak2- MtCmaxTmaxP2(i,1))/(MaxMak2- MinMak2);
fTmax2(i,j)=(MaxTard2- MtCmaxTmaxP2(i,2))/(MaxTard2- MinTard2);
```

```
U2(i,j)= W1*fCmax2(i,1)+ W2*fTmax2(i,1);
end
search2=find(Perm2(:,1)==0);
if search2>0;
   U2(search2)=0;
   Up2=U2;
else
   Up2=U2;
end
MaxU2=max(Up2);
k= find(MaxU2==Up2);
Px2= Pad2(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padre2=Px2(L,:);
   end
end
LT= [k(L,1)];
Ltb3=Perm2(LT,:);
Ltb4=[Ltb3(1,2), Ltb3(1,1)];
Listatabu2=[Listatabu1; Ltb3; Ltb4];
% %%%%3
for i=1:n;
   j=1:N;
   Padre2=Px2(L,:);
   t=randperm(N,2);
    if t(1,1)==Ltb1(1,1)&t(1,2)==Ltb1(1,2);
       t(1,:)=[0];
    elseif t(1,1)==Ltb2(1,1)&t(1,2)==Ltb2(1,2);
t(1,:)=[0];
 elseif t(1,1)==Ltb3(1,1)&t(1,2)==Ltb3(1,2);
t(1,:)=[0];
elseif t(1,1)==Ltb4(1,1)&t(1,2)==Ltb4(1,2);
t(1,:)=[0];
    end
     B1=find(t(1,1)==Padre2);
   B2=find(t(1,2)==Padre2);
   Padre2(1,B1)=t(1,2);
   Padre2(1,B2)=t(1,1);
    Pad3(i,j)=Padre2(:,j);

 for j= 1:2;
      Perm3(i,j)= t(:,j);
```

```
 end
end
for i= 1:n;
 [CmaxTmaxP3]= [KQFFSfitness(PT,Pad3(i,:),N,ME,Dd)];
 for j=1:2;
MtCmaxTmaxP3(i,j)= CmaxTmaxP3(:,j);
 end
end
 for i=1:n;
    j=1;
 Makespan max3(i,j)= MtCmaxTmaxP3(i,1);
 Makespan min3(i,j)= MtCmaxTmaxP3(i,1);
 Tardanzamax3(i,j)= MtCmaxTmaxP3(i,2);
 Tardanzamin3(i,j)= MtCmaxTmaxP3(i,2);
end
search=find(Perm3(:,1)==0);
if search>0;
   Makespan max3(search)=0;
   MaxMak3= max(Makespan max3);
   Tardanzamax3(search)=0;
   MaxTard3= max(Tardanzamax3);
   Makespan min3(search)=A;
   MinMak3=min(Makespan min3);
   Tardanzamin3(search)=A;
   MinTard3= min(Tardanzamin3);
else
  MaxMak3= max(Makespan max3);
  MinMak3= min(Makespan max3);
  MaxTard3= max(Tardanzamax3);
  MinTard3= min(Tardanzamax3);
end
if MaxTard3==MinTard3
   j=1;
   Cmaxx3=MtCmaxTmaxP3(:,j);
   if search>0;
      Cmaxx3(search)=A;
   end
   MinCmaxx3=min(Cmaxx3);
   k= find(MinCmaxx3==Cmaxx3);
Px3= Pad3(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padre3=Px3(L,:);
```

```
end
    if MaxMak3==MinMak3;
   j=2;
   Tmaxx3=MtCmaxTmaxP3(:,j);
   if search>0;
      Tmaxx3(search)=A;
   end
   MinTmaxx3=min(Tmaxx3);
   k= find(MinTmaxx3==Tmaxx3);
Px3= Pad3(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padre3=Px3(L,:);
 end
if MaxTard3~=MinTard3;
   if MaxMak3~=MinMak3;
for i=1:n;
   j=1;
% Normalizamos funciones de Makespan  y Tardanza:
fCmax3(i,j)=(MaxMak3- MtCmaxTmaxP3(i,1))/(MaxMak3- MinMak3);
fTmax3(i,j)=(MaxTard3- MtCmaxTmaxP3(i,2))/(MaxTard3- MinTard3);
U3(i,j)= W1*fCmax3(i,1)+ W2*fTmax3(i,1);
end
search2=find(Perm3(:,1)==0);
if search2>0;
   U3(search2)=0;
   Up3=U3;
else
   Up3=U3;
end
MaxU3=max(Up3);
k= find(MaxU3==Up3);
Px3= Pad3(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padre3=Px3(L,:);
   end
end
LT= [k(L,1)];
Ltb5=Perm3(LT,:);
Ltb6=[Ltb5(1,2), Ltb5(1,1)];
Listatabu3=[Listatabu2; Ltb5; Ltb6];
```

```
% Luego de obtener la lista tabu inicial, se pueden realizar n interacciones
% y se actualiza la lista tabu y el padre en cada corrida
for w=1:Wts;
   Padrepp=Padre3;
   Ltbf=Listatabu3;
for i=1:n;
   j=1:N;
   t=randperm(N,2);
    if t(1,1)==Ltbf(1,1)&t(1,2)==Ltbf(1,2);
       t(1,:)=[0];
    elseif t(1,1)==Ltbf(2,1)&t(1,2)==Ltbf(2,2);
t(1,:)=[0];
 elseif t(1,1)==Ltbf(3,1)&t(1,2)==Ltbf(3,2);
t(1,:)=[0];
elseif t(1,1)==Ltbf(4,1)&t(1,2)==Ltbf(4,2);
t(1,:)=[0];
elseif t(1,1)==Ltbf(5,1)&t(1,2)==Ltbf(5,2);
t(1,:)=[0];
elseif t(1,1)==Ltbf(6,1)&t(1,2)==Ltbf(6,2);
t(1,:)=[0];
    end
     B1=find(t(1,1)==Padrepp);
   B2=find(t(1,2)==Padrepp);
   Padrepp(1,B1)=t(1,2);
   Padrepp(1,B2)=t(1,1);
    Pad4(i,j)=Padrepp(:,j);

 for j= 1:2;
      Perm4(i,j)= t(:,j);
 end
end
for i= 1:n;
 [CmaxTmaxF]= [KQFFSfitness(PT,Pad4(i,:),N,ME,Dd)];
 for j=1:2;
MtCmaxTmaxF(i,j)= CmaxTmaxF(:,j);
 end
end
 for i=1:n;
    j=1;
 Makespan maxF(i,j)= MtCmaxTmaxF(i,1);
 Makespan minF(i,j)= MtCmaxTmaxF(i,1);
 TardanzamaxF(i,j)= MtCmaxTmaxF(i,2);
 TardanzaminF(i,j)= MtCmaxTmaxF(i,2);
end
```

```
search=find(Perm4(:,1)==0);
if search>0;
   Makespan maxF(search)=0;
   MaxMakF= max(Makespan maxF);
   TardanzamaxF(search)=0;
   MaxTardF= max(TardanzamaxF);
   Makespan minF(search)=A;
   MinMakF=min(Makespan minF);
   TardanzaminF(search)=A;
   MinTardF= min(TardanzaminF);
else
  MaxMakF= max(Makespan maxF);
  MinMakF= min(Makespan maxF);
  MaxTardF= max(TardanzamaxF);
  MinTardF= min(TardanzamaxF);
end
if MaxTardF==MinTardF;
   j=1;
   CmaxxF=MtCmaxTmaxF(:,j);
   if search>0;
      CmaxxF(search)=A;
   end
   MinCmaxxF=min(CmaxxF);
   k= find(MinCmaxxF==CmaxxF);
PF= Pad4(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
PadreF=PF(L,:);
Padre3=PadreF;
end
 if MaxMakF==MinMakF;
   j=2;
   TmaxxF=MtCmaxTmaxF(:,j);
   if search>0;
      TmaxxF(search)=A;
   end
   MinTmaxxF=min(TmaxxF);
   k= find(MinTmaxxF==TmaxxF);
PF= Pad4(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
PadreF=PF(L,:);
```

```
Padre3=PadreF;
 end
if MaxTardF~=MinTardF;
   if MaxMakF~=MinMakF;
for i=1:n;
   j=1;
% Normalizamos funciones de Makespan  y Tardanza:
fCmaxF(i,j)=(MaxMakF- MtCmaxTmaxF(i,1))/(MaxMakF- MinMakF);
fTmaxF(i,j)=(MaxTardF- MtCmaxTmaxF(i,2))/(MaxTardF- MinTardF);
U4(i,j)= W1*fCmaxF(i,1)+ W2*fTmaxF(i,1);
end
search2=find(Perm4(:,1)==0);
if search2>0;
   U4(search2)=0;
   UF=U4;
else
   UF=U4;
end
MaxUF=max(UF);
k= find(MaxUF==UF);
PF= Pad4(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
PadreF=PF(L,:);
Padre3=PadreF;
   end
end
LT= [k(L,1)];
Ltb7=Perm4(LT,:);
Ltb8=[Ltb7(1,2), Ltb7(1,1)];
Listatabu3(1,:)=[Ltbf(3,1),Ltbf(3,2)];
Listatabu3(2,:)=[Ltbf(4,1),Ltbf(4,2)];
Listatabu3(3,:)=[Ltbf(5,1),Ltbf(5,2)];
Listatabu3(4,:)=[Ltbf(6,1),Ltbf(6,2)];
Listatabu3(5,:)=[Ltb7(1,1), Ltb7(1,2)];
Listatabu3(6,:)=[Ltb8(1,1), Ltb8(1,2)];
j= 1:N;
Padres(w,j)=PadreF(:,j);
j=1:2;
CmaxTmaxPF(w,j)= KQFFSfitness(PT,Padres(w,:),N,ME,Dd);
% for j=1:2;
% MtCmaxTmaxPF(i,j)= CmaxTmaxPF(:,j)
% end
```

```
 end
MaxCTPF= max(CmaxTmaxPF);
MinCTPF= min(CmaxTmaxPF);
if MaxCTPF(1,2)==MinCTPF(1,2);
   j=1;
   CmaxxPF=CmaxTmaxPF(:,j);
   MinCmaxxPF=min(CmaxxPF);
   k= find(MinCmaxxPF==CmaxxPF);
PF= Padres(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
PadreFF=PF(L,:);
end
if MaxCTPF(1,1)==MinCTPF(1,1);
   j=2;
 TmaxxPF=CmaxTmaxPF(:,j);
   MinTmaxxPF=min(TmaxxPF);
   k= find(MinTmaxxPF==TmaxxPF);
PF= Padres(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
PadreFF=PF(L,:);
end
if MaxCTPF(1,2)~=MinCTPF(1,2);
if MaxCTPF(1,1)~=MinCTPF(1,1);
for i=1:w;
   j=1;
% Normalizamos funciones de Makespan  y Tardanza:
fCmaxPF(i,j)=(MaxCTPF(1,1)- CmaxTmaxPF(i,1))/(MaxCTPF(1,1)- MinCTPF(1,1));
fTmaxPF(i,j)=(MaxCTPF(1,2)- CmaxTmaxPF(i,2))/(MaxCTPF(1,2)- MinCTPF(1,2));
%Con la ecuación siguiente se determina cual de las secuencias da un
%mejor resultado. Se colocó un peso w del 0.4 para Makespan  y 0.6 para
%tardanza. La solución cuyo resultado sea mayor acercandose a uno, es la
%elegida como mejor secuencia.
UFF(i,j)= W1*fCmaxPF(i,1)+ W2*fTmaxPF(i,1);
end
MaxUFF=max(UFF);
k= find(MaxUFF==UFF);
PF= Padres(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
```

```
PadreFF=PF(L,:);
end
end
PadreTS=PadreFF;
CmaxTmaxPFF= [KQFFSfitness(PT,PadreTS,N,ME,Dd)];
end
```

Cruzamiento

```
function Desc= Crossover(parent1,parent2,N,ME,PT,Dd,W1,W2)
% Cruzamiento padres iniciales
A=2000000;
B=1000000;
N2=N/2-1;
N3= round(N2);
perm=randperm(N,1);
x=perm+randperm(N3,1);
if x>N;
   y=N;
else
   y=x;
end
Cross= (perm:y);
Cross1=parent1(Cross);
Cross2=parent2(Cross);
% parent1(perm:y)=Cross2
% parent2(perm:y)=Cross1
parent1(perm:y)=A;
parent2(perm:y)=A;
l=size(Cross);
l1=l(1,2);
for i=1:l1;
u= find(Cross2(1,i)==parent1);
if u>0;
   u1=u;
else
   u1=0;
end
j=1;
u2(i,j)= u1(:,j);
end
for i=1:l1;
   if u2(i,1)>0;
```

```
      parent1(1,u2(i,1))=B;
   end
end
parent1(perm:y)= Cross2;
sizeCross= size(Cross);
TamCross= sizeCross(1,2);
for i=1:TamCross;
   fcross= find(Cross1(1,i)==parent1);
   if fcross>0;
      rep= fcross;
   else
      rep=0;
   end
   j=1;
   repet(i,j)= rep(:,j);
end
for i=1:TamCross;
   if repet(i,1)==0;
      fcross1=find(B==parent1);
      pos1fcross1=fcross1(1,1);
      parent1(1,pos1fcross1)=Cross1(1,i);
   end
end
%%%Para parent2
for i=1:l1;
up2= find(Cross1(1,i)==parent2);
if up2>0;
   u1p2=up2;
else
   u1p2=0;
end
j=1;
u2p2(i,j)= u1p2(:,j);
end
for i=1:l1;
   if u2p2(i,1)>0;
      parent2(1,u2p2(i,1))=B;
   end
end
parent2(perm:y)= Cross1;
sizeCrossp2= size(Cross);
TamCrossp2= sizeCrossp2(1,2);
for i=1:TamCrossp2;
   fcrossp2= find(Cross2(1,i)==parent2);
```

```
   if fcrossp2>0;
      repp2= fcrossp2;
   else
      repp2=0;
   end
   j=1;
   repetp2(i,j)= repp2(:,j);
end
for i=1:TamCrossp2;
   if repetp2(i,1)==0;
      fcross1p2=find(B==parent2);
      pos1fcross1p2=fcross1p2(1,1);
      parent2(1,pos1fcross1p2)=Cross2(1,i);
   end
end
Hijos= [parent1; parent2];
for i= 1:2;
 [CmaxTmaxH]= [KQFFSfitness(PT,Hijos(i,:),N,ME,Dd)];
 for j=1:2;
MtCmaxTmaxH(i,j)= CmaxTmaxH(:,j);
end
end
MaxCTH= max(MtCmaxTmaxH);
MinCTH= min(MtCmaxTmaxH);
if MtCmaxTmaxH(1,1)==MtCmaxTmaxH(2,1);
   if  MtCmaxTmaxH(1,2)>MtCmaxTmaxH(2,2);
      Hijo= Hijos(2,:);
      CmaxTmaxHijo= MtCmaxTmaxH(2,:);
   else
       Hijo= Hijos(1,:);
       CmaxTmaxHijo= MtCmaxTmaxH(1,:);
   end
else
if MtCmaxTmaxH(1,2)==MtCmaxTmaxH(2,2);
   if  MtCmaxTmaxH(1,1)>MtCmaxTmaxH(2,1);
      Hijo= Hijos(2,:);
      CmaxTmaxHijo= MtCmaxTmaxH(2,:);
   else
       Hijo= Hijos(1,:);
       CmaxTmaxHijo= MtCmaxTmaxH(1,:);
   end
end
end
if MtCmaxTmaxH(1,1)~=MtCmaxTmaxH(2,1);
```

```
   if MtCmaxTmaxH(1,2)~=MtCmaxTmaxH(2,2);
% else
for i=1:2;
   j=1;
fCmaxH(i,j)=(MaxCTH(1,1)- MtCmaxTmaxH(i,1))/(MaxCTH(1,1)- MinCTH(1,1));
fTmaxH(i,j)=(MaxCTH(1,2)- MtCmaxTmaxH(i,2))/(MaxCTH(1,2)- MinCTH(1,2));
UH(i,j)= W1*fCmaxH(i,1)+ W2*fTmaxH(i,1);
end
MaxUH=max(UH);
k= find(MaxUH==UH);
K= size(k);
l= K(1,1);
if l>1;
L= randi(l);
Hijo= Hijos(L,:);

else
   Hijo=Hijos(k,:);
end
   end
end
Desc= Hijo;
End
```

Mutación

```
function Mut= Mutacion(Hijo,PT,N,ME,Dd, W1, W2)
%Con esta función se somete a mutación el hijo obtenido del cruzamiento,
%intercambiando dos posiciones dentro de la secuencia hijo. Además evalúa
%tanto al hijo como a la mutación y escoge cuál de los dos es la mejor
%solución
HijoM= [Hijo];
N2=N/2-1;
N3= round(N2);
perm=randperm(N3,1);
x=perm+randperm(N3,1);
if x>N;
   y=N;
else
   y=x;
end
posmix= (perm:y);
comb= HijoM(perm:y);
```

```
l= size(comb);
L=l(1,2);
mix=randperm(L);
comb2=comb(:,mix);
HijoM(perm:y)=comb2;

%%%
CtmaxtmaxHijo= [KQFFSfitness(PT,Hijo,N,ME,Dd)];
CtmaxtmaxHijoM= [KQFFSfitness(PT,HijoM,N,ME,Dd)];
MtCmaxTmaxH= [CtmaxtmaxHijo; CtmaxtmaxHijoM];
Hijos=[Hijo; HijoM];
MaxCTH= max(MtCmaxTmaxH);
MinCTH= min(MtCmaxTmaxH);
if MtCmaxTmaxH(1,1)==MtCmaxTmaxH(2,1);
   if  MtCmaxTmaxH(1,2)>MtCmaxTmaxH(2,2);
      Hijo= Hijos(2,:);
      CmaxTmaxHijo= MtCmaxTmaxH(2,:);
   else
       Hijo= Hijos(1,:);
       CmaxTmaxHijo= MtCmaxTmaxH(1,:);
   end
else
if MtCmaxTmaxH(1,2)==MtCmaxTmaxH(2,2);
   if  MtCmaxTmaxH(1,1)>MtCmaxTmaxH(2,1);
      Hijo= Hijos(2,:);
      CmaxTmaxHijo= MtCmaxTmaxH(2,:);
   else
       Hijo= Hijos(1,:);
       CmaxTmaxHijo= MtCmaxTmaxH(1,:);
   end
end
end
if MtCmaxTmaxH(1,1)~=MtCmaxTmaxH(2,1);
   if MtCmaxTmaxH(1,2)~=MtCmaxTmaxH(2,2);
for i=1:2;
   j=1;
fCmaxH(i,j)=(MaxCTH(1,1)- MtCmaxTmaxH(i,1))/(MaxCTH(1,1)- MinCTH(1,1));
fTmaxH(i,j)=(MaxCTH(1,2)- MtCmaxTmaxH(i,2))/(MaxCTH(1,2)- MinCTH(1,2));
UH(i,j)= W1*fCmaxH(i,1)+ W2*fTmaxH(i,1);
end
MaxUH=max(UH);
k= find(MaxUH==UH);
K= size(k);
l= K(1,1);
```

```
if l>1;
L= randi(l);
Hijo= Hijos(L,:);

else
   Hijo=Hijos(k,:);
end
   end
end
Mut= Hijo;
```

Cj secuencia final

```
function [Cjf]= Cjsecfinal (PT,P,N,ME,Dd)
% Con esta función se calcula el tiempo de finalización de cada trabajo
% % % para una secuencia determinada
 [A B]=size(ME);
Sum=0;
X=PT(P,:);
Dds= Dd(P,:);
%%%
 MEIn=ME(1,1);
   R(1,:)=[1 MEIn];
   for k=2:B;
      j=1:2;
      MEI=MEIn+1;
      MEF= MEI+ME(1,k)-1;
      MEIn=MEF;
      RR=[MEI MEF];
      R(k,j)=RR(:,j);
   end
    for i=1:B;
      Sum=Sum+ME(1,i);
    end
   for i=1:N;
      j=1:Sum;
      E1(i,j)=0;
   end
   %%%%Para trabajo 1 según la secuencia
   Xi=0;
   for i=1:B;
      R1=R(i,1);
      E1(1,R1)=Xi+X(1,i);
```

```
    Xi=E1(1,R1);
  end
  %%%% Para trabajos 2 hasta N en etapa 1
  for i=2:N;
    j=1:ME(1,1);
  E2= E1(i-1,j);
  MinE2= min(E2);
  FE1= find(MinE2==E2);
  SFE1=size(FE1);
  K=SFE1(1,2);
  l=randi(K);
  F=FE1(1,l);
E1(i,j)= E1(i-1,j);
E1(i,F)= E1(i-1,F)+X(i,1);
  end
for i=2:N;
  E2=E1(i,:);
  for j=2:B;
  E2X= E1(i-1,R(j,1):R(j,2));
  MinE2X= min(E2X);
 FER= find(MinE2X==E2X);
 SFER=size(FER);
 K=SFER(1,2);
 l=randi(K);
F=FER(1,l);
EXB=E1(i,R(j-1,1):R(j-1,2));
EXA=E1(i-1,R(j-1,1):R(j-1,2));
YX=EXB==EXA;
FYX=find(0==YX);
CjA=EXB(1,FYX);
CjB=E2X(1,F);
if CjA>CjB;
   E2X(1,F)=CjA+X(i,j);
else
   E2X(1,F)=CjB+X(i,j);
end
E1(i,R(j,1):R(j,2))=E2X;
  end
end
Cjf=E1;
%  NewCj= 'CjN';
% ExcRT=xlswrite(NewCj,Cjf,1);
End
```

Algoritmo genético – Cuerpo principal

```
clc, clear all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Este algoritmo genetico tiene como objetivo resolver problemas de tipo
%flexible flow shop de dos objetivos que son Makespan  y tardanza maxima.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%La población inicial se realiza de manera aleatoria
%para elegir los padres
TP= 'Pruebazap';
PT= xlsread(TP,1);%Tiempo de procesamiento de cada trabajo
%en cada máquina
S= size(PT); %Tamaño de la matriz
N= S(1,1);%trabajos
E= S(1,2);%Estaciones
ME=[3 1 5 3];%# de máquinas por estaciones
Dd= xlsread(TP,2);% Tiempo maximo de entrega por trabajo
SecProg= [9    4    12    18    10    16    7    14    5    1    6    20    3    17    8    11    2    15    19
13];
WAg=100; %# de iteracciones para AGgg
Wts=20;%#de iteracciones en Tabu Search
W1=0.1;%Peso dado a Makespan
W2=0.9;%Peso dado a Tardanza
ProbM= 0.4; %Probabilidad de mutación
p= 10; %tamaño de la población inicial
%%%Se generan 2 poblaciones aleatorias y se escoge la mejor de cada una
%%%para luego ser sometidos a Tabu Search y de esta forma obtener los
%%%dos padres iniciales para el cruzamiento.
Pob1= Generar_pobl(N,p);%%Población1
for i= 1:p;
 [CmaxTmax1]= [KQFFSfitness(PT,Pob1(i,:),N,ME,Dd)];%Calculo del fitness
 %%%tiene dos variables de salida, Makespan  y tardanza
 for j=1:2;
MtCmaxTmax(i,j)= CmaxTmax1(:,j);%%Matriz de fitness para población 1
end
end
%%%Se evalua cual secuencia es mejor dentro de la población 1
%%En caso de que todas las secuencias tengan como resultado la misma
%%tardanza, se escoge la secuencia con menor Cmax.
MaxCT= max(MtCmaxTmax);
MinCT= min(MtCmaxTmax);
if MaxCT(1,2)==MinCT(1,2);
  j=1;
    CmaxPini1=MtCmaxTmax(:,j);
```

```
   MinCmaxPini1=min(CmaxPini1);
   k= find(MinCmaxPini1==CmaxPini1);
Px1= Pob1(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padini1=Px1(L,:);
end
%%En caso de que todas las secuencias tengan como resultado el mismo
%%Makespan , se escoge la secuencia con menor Tardanza.
MaxCT= max(MtCmaxTmax);
if MaxCT(1,1)==MinCT(1,1);
   j=2;
   TmaxPini1=MtCmaxTmax(:,j);
   MinTmaxPini1=min(TmaxPini1);
   k= find(MinTmaxPini1==TmaxPini1);
Px1= Pob1(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padini1=Px1(L,:);
end
%%%En caso de que las secuencias tengan diferentes Makespan  se utiliza la
%%%funcion de normalizacion cuando hay mas de un objetivo, los pesos w1 y
%%%w2 los determina quien maneja el programa de producción
if MaxCT(1,2)~=MinCT(1,2);
if MaxCT(1,1)~=MinCT(1,1);
for i=1:p;
   j=1;
fCmax(i,j)=(MaxCT(1,1)- MtCmaxTmax(i,1))/(MaxCT(1,1)- MinCT(1,1));
fTmax(i,j)=(MaxCT(1,2)- MtCmaxTmax(i,2))/(MaxCT(1,2)- MinCT(1,2));
U(i,j)= W1*fCmax(i,1)+ W2*fTmax(i,1);
end
MaxU=max(U);
k= find(MaxU==U);
Px1= Pob1(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padini1=Px1(L,:);
end
end
%%%%%%%%%
%Generamos Población 2 para obtener el Padre #2 con Tabu Search, esto se realiza
```

```
%igual que en Población 1.
Pob2= Generar_pobl(N,p);
for i= 1:p;
 [CmaxTmax2]= [KQFFSfitness(PT,Pob2(i,:),N,ME,Dd)];
 for j=1:2;
MtCmaxTmax2(i,j)= CmaxTmax2(:,j);
end
end
MaxCT2= max(MtCmaxTmax2);
MinCT2= min(MtCmaxTmax2);
if MaxCT2(1,2)==MinCT2(1,2);
  j=1;
    CmaxPini2=MtCmaxTmax2(:,j);
    MinCmaxPini2=min(CmaxPini2);
    k= find(MinCmaxPini2==CmaxPini2);
Px2= Pob2(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padini2=Px2(L,:);
end
if MaxCT2(1,1)==MinCT2(1,1);
    j=2;
    TmaxPini2=MtCmaxTmax2(:,j);
    MinTmaxPini2=min(TmaxPini2);
    k= find(MinTmaxPini2==TmaxPini2);
Px2= Pob2(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Padini2=Px2(L,:);
end
if MaxCT2(1,2)~=MinCT2(1,2);
if MaxCT2(1,1)~=MinCT2(1,1);
for i=1:p;
    j=1;
fCmax2(i,j)=(MaxCT2(1,1)- MtCmaxTmax2(i,1))/(MaxCT2(1,1)- MinCT2(1,1));
fTmax2(i,j)=(MaxCT2(1,2)- MtCmaxTmax2(i,2))/(MaxCT2(1,2)- MinCT2(1,2));
U2(i,j)= W1*fCmax2(i,1)+ W2*fTmax2(i,1);
end
MaxU2=max(U2);
k= find(MaxU2==U2);
Px2= Pob2(k,:);
K= size(k);
```

```
l= K(1,1);
L= randi(l);
Padini2=Px2(L,:);
end
end
for i=1:WAg;
   %%%%%%%%%%%%%
   %Tabu Search%
   %%%%%%%%%%%%%
%Utilizamos busqueda tabu para buscar el Padre #1 con
%secuencia inicial=Padini1, de igual forma para el Padre #2
%usando como secuencia inicial=Padini2;
Parent1= TsearchFFS(Padini1,N,ME,PT,Dd,Wts,W1,W2);
Parent2= TsearchFFS(Padini2,N,ME,PT,Dd,Wts,W1,W2);

   %%%%%%%%%%%%%
   %Cruzamiento%
   %%%%%%%%%%%%%
%Realizamos cruce de los padres desde una posición A hasta una posición B,
%el tamaño entre A  B es maximo la mitad del cromosoma. Estas posiciones
%se buscan en el padre 1 y se cruzan con el padre 2 y viceversa, de manera
%que nos da como resultados dos hijos, se elige el mejor de estos dos.
BestHijoCruce= Crossover(Parent1,Parent2,N,ME,PT,Dd,W1,W2);
CmaxTmaxBHCruce= [KQFFSfitness(PT,BestHijoCruce,N,ME,Dd)];
BestHijoFinal= BestHijoCruce;
CmaxTmaxBHfinal=CmaxTmaxBHCruce;
   %%%%%%%%%%%%%
   %Mutación%
   %%%%%%%%%%%%%
%%Para este proceso la mejor secuencia dada del cruzamiento, se puede
%%someter a mutación según una probabilidad que determina quien ejecuta el
%%programa, donde se escoge de manera aleatoria de una posicion A a una B,
%%el tamaño entre A y B es máximo la mitad del cromosoma, esta seccion del
%%cromosoma se somete a cambios aleatorias de las posiciones.
pm= rand(1);
 if pm<ProbM;
BestHijoFinal= Mutacion(BestHijoCruce,PT,N,ME,Dd,W1,W2);
 end
%%%%%%%
%Se obtiene el consolidado de las mejores secuencias y se calcula el fitness
%de cada una de ellas.
j=1:N;
MtBesthijo(i,j)=BestHijoFinal(:,j);
j=1:2;
```

```
MTctmaxtmaxBhijo(i,j)= KQFFSfitness(PT,MtBesthijo(i,:),N,ME,Dd);
end
%%%% Si las secuencias finales tienen la misma tardanza, se escoge la de
%%%% menor Makespan , si hay más de dos con el mejor resultado, se elige una
%%%% secuencia de manera aleatoria
MaxCT= max(MTctmaxtmaxBhijo);
MinCT= min(MTctmaxtmaxBhijo);
mm1=MaxCT(1,2)==MinCT(1,2);
mm2=MaxCT(1,1)==MinCT(1,1);
mm3=MaxCT(1,2)~=MinCT(1,2);
mm4=MaxCT(1,1)~=MinCT(1,1);
if MaxCT(1,2)==MinCT(1,2);%Tardanza
   j=1;
   Cmaxx=MTctmaxtmaxBhijo(:,j);
   MinCmaxx=min(Cmaxx);
   k= find(MinCmaxx==Cmaxx);
BH= MtBesthijo(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
BestHijoSol= BH(L,:)%Mejor secuencia del AG ó:
end
%%%% Si las secuencias finales tienen el mismo Makespan , se escoge la de
%%%% menor tardanza, si hay más de dos con el mejor resultado, se elige una
%%%% secuencia de manera aleatoria
   if MaxCT(1,1)==MinCT(1,1);%Makespan
 j=2;
   Tmaxx=MTctmaxtmaxBhijo(:,j);
   MinTmaxx=min(Tmaxx);
   k= find(MinTmaxx==Tmaxx);
BH= MtBesthijo(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
BestHijoSol= BH(L,:);%Mejor secuencia del AG ó:
   end
%%%%%
%En caso de que las secuencias tengan diferentes fitness, se utiliza la
%función de normalización para funciones multiobjetivos, los pesos w1 y w2
%son determinados por quién maneja la programación
if MaxCT(1,2)~=MinCT(1,2);
   if MaxCT(1,1)~=MinCT(1,1);
for i=1:WAg;
   j=1;
```

```
fCmax(i,j)=(MaxCT(1,1)- MTctmaxtmaxBhijo(i,1))/(MaxCT(1,1)- MinCT(1,1));
fTmax(i,j)=(MaxCT(1,2)- MTctmaxtmaxBhijo(i,2))/(MaxCT(1,2)- MinCT(1,2));
UBestHijo(i,j)= W1*fCmax(i,1)+ W2*fTmax(i,1);
end
MaxUBestHijo=max(UBestHijo);
k= find(MaxUBestHijo==UBestHijo);
BH= MtBesthijo(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
BestHijoSol= BH(L,:);%Mejor secuencia del AG
  end
end
SolBestHijo=BestHijoSol
CmaxTmaxBHSol=[KQFFSfitness(PT,BestHijoSol,N,ME,Dd)]%Mejor fitness del AG
CjBestHijo= Cjsecfinal(PT,BestHijoSol,N,ME,Dd)
CJF= 'CJFreprog';
ExcCJ=xlswrite(CJF,CjBestHijo,1)
```

3. Modelo de reprogramación de la producción

Cálculo del fitness- Diferencia de los tiempos de inicio y tardanza total

```
function [DiffTiTmax]= RKQFFSfitness (PT,N,ME,Dd,Bd,yk,SecProg,P,CjSec,CjNi,RT)
% % % % Con esta función se calcula el tiempo de finalización de cada trabajo
% % % % para una secuencia determinada
NSecProg=SecProg;
NSecProg(Bd:N)=P;
[A B]=size(ME);
Sum=0;
X=PT(NSecProg,:); % Tiempos de procesamiento ordenado segun nueva secuencia
Dds= Dd(NSecProg,:);% Tiempos de entrega ordenado segun nueva secuencia
Bx= SecProg(1,Bd);
%RT1=RT;
RT(Bx,1)=yk;
Reltm= RT(NSecProg,:);%% Release time ordenado segun nueva secuencia
%%%% Calculo de Cj para la nueva secuencia
CJRP= KqCj (PT,SecProg,P,N,Bd,ME,Dd,CjNi,CjSec, Reltm);
%%%Tiempo de inicio para cada trabajo obtenido del calculo de Cj
Sum=0;
 for i=1:B;
      Sum=Sum+ME(1,i);
 end
 MENi= Sum-ME(1,B)+1;
```

```
 MENf=Sum;
for i=2:N;
   j=1:ME(1,1);
   XCjsec=CJRP(i,j)~=CJRP(i-1,j);
   FCjini= find(XCjsec==1);
   Cjini=CJRP(i,FCjini)-X(i,1);
j=1;
CjNRP(i,j)= Cjini(:,j);
end
%%%Diferencia entre los tiempos de inicio de la secuencia inicial y la
%%%nueva programación
Tsecp= transpose(SecProg);
TNsecp= transpose(NSecProg);
for i=1:N;
   j=1;
Fsecp= find(Tsecp(i,j)==TNsecp);
for j=1;
   Fsecpx(i,j)= Fsecp(:,j);
end
end
for i=1:N;
   CJNRP2= CjNRP(Fsecpx,1);
end
CjiniRp=CJNRP2;
for i=1:N;
DifCJi(i,j)= CJNRP2(i,1)-CjNi(i,1);
end
AbsCJi=abs(DifCJi);
DiffTi=[sum(AbsCJi)];
%%%Calculo de la tardanza para la nueva secuencia
 MEIn=ME(1,1);
   R(1,:)=[1 MEIn];
   for k=2:B;
      j=1:2;
      MEI=MEIn+1;
      MEF= MEI+ME(1,k)-1;
      MEIn=MEF;
      RR=[MEI MEF];
      R(k,j)=RR(:,j);
   end
EP=R(B,1):R(B,2);
s=size(EP);
sf=s(1,2);
 for i=1:N;
```

```
  CjE=CJRP(i,EP);
  j=1:sf;
  CjEF(i,j)=CjE(:,j);
 end
 CJi(1)=max(CjEF(1,:));
  [s1 s2]=size(CjEF);
  for i=2:N;
     BCj=CjEF(i-1,:)==CjEF(i,:);
     FBCj=find(0==BCj);
     CJi(i)=CjEF(i,FBCj);
  end
    CJiF=transpose(CJi);
%     MCJiF=max(CJiF)
%     FMCJiF= find(MCJiF==CJiF)
for i=1:N;
   Tardxj= CJiF-Dds;
   if Tardxj(i,1)>0;
      Tardanza(i,1)=Tardxj(i,1);
   else
      Tardanza(i,1)=0;
   end
   TardT= transpose(Tardanza);
end
%%%% Diferencia entre los tiempos iniciales de las secuencias y tardanza de
%%%% la nueva secuencia
MaxCmax=max(CJiF);
DiffTiTmax= [DiffTi, sum(TardT),MaxCmax];
```

Tabu Search

```
function PadreTS= RTsearchFFS(padrex,PT,N,ME,Dd,Bd,yk,SecProg,CjSec,CjNi,Wts,W1,W2,RT)
%%%%%%Función para realizar búsqueda tabu en la secuencia de reprogramación
secini= padrex;
[SA SB]= size(secini);
n=10;
A=1000000;
if SB<=3;
   PadreTS= padrex;
else
%%%%%Padre1
for i=1:n;
   j=1:SB;
   Padrex1= [secini];
```

```
   tx=randperm(SB,2);%Posiciones en secini a cruzar
   B1=Padrex1(1,tx(1,1));
   B2=Padrex1(1,tx(1,2));
   t= [B1 B2];
   Padrex1(1,tx(1,1))=B2;
   Padrex1(1,tx(1,2))=B1;
%    Padreini1= Padrex1
   Padre1(i,j)=Padrex1(:,j);
 for j= 1:2;
      Perm(i,j)= t(:,j);
 end
end
for i= 1:n;
TsP1= RKQFFSfitness (PT,N,ME,Dd,Bd,yk,SecProg,Padre1(i,:),CjSec,CjNi,RT);
for j=1:3;
   MatTsP1(i,j)= TsP1(:,j);
end
end
 MatTsP11=MatTsP1(:,1);
 MatTsP12=MatTsP1(:,2);
%%%
MaxDti1= max(MatTsP11);
MinDti1= min(MatTsP11);
MaxTard1=max(MatTsP12);
MinTard1=min(MatTsP12);
Xp11=MaxDti1==MinDti1;
if MaxDti1==MinDti1;
   MinTardanza1=min(MatTsP12);
   k= find(MinTardanza1==MatTsP12);
Px1= Padre1(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Pad1=Px1(L,:);
end
Xp12=MaxTard1==MinTard1;
if MaxTard1==MinTard1;
   MinDifti1=min(MatTsP11);
   k= find(MinDifti1==MatTsP11);
Px1= Padre1(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Pad1=Px1(L,:);
```

```
end
Xp13=MaxDti1~=MinDti1;
Xp14=MaxTard1~=MinTard1;
if MaxDti1~=MinDti1;
   if MaxTard1~=MinTard1;
for i=1:n;
   j=1;
% Normalizamos funciones de Makespan  y Tardanza:
fDifti1(i,j)=(MaxDti1-MatTsP11(i,1))/(MaxDti1-MinDti1);
fTmax1(i,j)=(MaxTard1- MatTsP12(i,1))/(MaxTard1- MinTard1);
%Con la ecucación siguiente se determina cual de las secuencias da un
%mejor resultado. Se colocó un peso w del 0.4 para Makespan  y 0.6 para
%tardanza. La solución cuyo resultado sea mayor acercandose a uno, es la
%elegida como mejor secuencia.
U(i,j)= W1*fDifti1(i,1)+ W2*fTmax1(i,1);
end
MaxU=max(U);
k= find(MaxU==U);
Px1= Padre1(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Pad1=Px1(L,:);
   end
end
LT= [k(L,1)];
Ltb1=Perm(LT,:);
Ltb2=[Ltb1(1,2), Ltb1(1,1)];
Listatabu1=[Ltb1; Ltb2];
%%%%% Padre 2
for i=1:n;
   j=1:SB;
   tx=randperm(SB,2);%Posiciones en secini a cruzar
   B1=Pad1(1,tx(1,1));
   B2=Pad1(1,tx(1,2));
   t= [B1 B2];
    if t(1,1)==Ltb1(1,1)&t(1,2)==Ltb1(1,2);
       t(1,:)=[0];
    elseif t(1,1)==Ltb2(1,1)&t(1,2)==Ltb2(1,2);
t(1,:)=[0];
    end
     BB1=find(t(1,1)==Pad1);
   BB2=find(t(1,2)==Pad1);
   Pad1(1,BB1)=t(1,2);
```

```
   Pad1(1,BB2)=t(1,1);
    Padre2(i,j)=Pad1(:,j);
    for j= 1:2;
      Perm2(i,j)= t(:,j);
 end
end
for i= 1:n;
TsP2= RKQFFSfitness (PT,N,ME,Dd,Bd,yk,SecProg,Padre2(i,:),CjSec,CjNi,RT);
for j=1:3;
   MatTsP2(i,j)= TsP2(:,j);
end
end
 MatTsP21=MatTsP2(:,1);
 MatTsP22=MatTsP2(:,2);
 MatTsP21N=MatTsP21;
 MatTsP22N=MatTsP22;
%%%
searchx=find(Perm2(:,1)==0);
if searchx>0;
 MatTsP21(searchx)=0;
 MaxDti2= max(MatTsP21);
 MatTsP21(searchx)=A;
 MinDti2= min(MatTsP21);
 MatTsP22(searchx)=0;
 MaxTard2=max(MatTsP22);
 MatTsP22(searchx)=A;
 MinTard2=min(MatTsP22);
else
MaxDti2= max(MatTsP21);
MinDti2= min(MatTsP21);
MaxTard2=max(MatTsP22);
MinTard2=min(MatTsP22);
end
X21=MaxDti2==MinDti2;
if MaxDti2==MinDti2;
   MinTardanza2=min(MatTsP22);
   k= find(MinTardanza2==MatTsP22);
Px2= Padre2(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Pad2=Px2(L,:);
end
X22=MaxTard2==MinTard2;
```

```
 if MaxTard2==MinTard2;
   MinDifti2=min(MatTsP21);
   k= find(MinDifti2==MatTsP21);
Px2= Padre2(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Pad2=Px2(L,:);
 end
 X23=MaxTard2~=MinTard2;
 X24=MaxDti2~=MinDti2;
if MaxTard2~=MinTard2;
   if MaxDti2~=MinDti2;
for i=1:n;
   j=1;
% Normalizamos funciones de Makespan  y Tardanza:
fDifti2(i,j)=(MaxDti2- MatTsP21N(i,1))/(MaxDti2- MinDti2);
fTmax2(i,j)=(MaxTard2- MatTsP22N(i,1))/(MaxTard2- MinTard2);
U2(i,j)= W1*fDifti2(i,1)+ W2*fTmax2(i,1);
end
if searchx>0;
   U2(searchx)=0;
   Up2=U2;
else
   Up2=U2;
end
MaxU2=max(Up2);
k= find(MaxU2==Up2);
Px2= Padre2(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Pad2=Px2(L,:);
end
end
LT= [k(L,1)];
Ltb3=Perm2(LT,:);
Ltb4=[Ltb3(1,2), Ltb3(1,1)];
Listatabu2=[Listatabu1; Ltb3; Ltb4];
%%%%%Padre 3
for i=1:n;
   j=1:SB;
tx=randperm(SB,2);%Posiciones en secini a cruzar
   B1=Pad2(1,tx(1,1));
```

```
   B2=Pad2(1,tx(1,2));
   t= [B1 B2];
    if t(1,1)==Ltb1(1,1)&t(1,2)==Ltb1(1,2);
       t(1,:)=[0];
    elseif t(1,1)==Ltb2(1,1)&t(1,2)==Ltb2(1,2);
t(1,:)=[0];
 elseif t(1,1)==Ltb3(1,1)&t(1,2)==Ltb3(1,2);
t(1,:)=[0];
elseif t(1,1)==Ltb4(1,1)&t(1,2)==Ltb4(1,2);
t(1,:)=[0];
    end
   BB1=find(t(1,1)==Pad2);
   BB2=find(t(1,2)==Pad2);
   Pad2(1,BB1)=t(1,2);
   Pad2(1,BB2)=t(1,1);
   Padre3(i,j)=Pad2(:,j);
    for j= 1:2;
      Perm3(i,j)= t(:,j);
    end
end
for i= 1:n;
TsP3= RKQFFSfitness (PT,N,ME,Dd,Bd,yk,SecProg,Padre3(i,:),CjSec,CjNi,RT);
for j=1:3;
   MatTsP3(i,j)= TsP3(:,j);
end
end
 MatTsP31=MatTsP3(:,1);
 MatTsP32=MatTsP3(:,2);
 MatTsP31N=MatTsP31;
 MatTsP32N=MatTsP32;
%%%
searchx=find(Perm3(:,1)==0);
if searchx>0;
 MatTsP31(searchx)=0;
 MaxDti3= max(MatTsP31);
 MatTsP31(searchx)=A;
 MinDti3= min(MatTsP31);
 MatTsP32(searchx)=0;
 MaxTard3=max(MatTsP32);
 MatTsP32(searchx)=A;
 MinTard3=min(MatTsP32);
else
MaxDti3= max(MatTsP31);
MinDti3= min(MatTsP31);
```

```
MaxTard3=max(MatTsP32);
MinTard3=min(MatTsP32);
end
X31=MaxDti3==MinDti3;
if MaxDti3==MinDti3;
   MinTardanza3=min(MatTsP32);
   k= find(MinTardanza3==MatTsP32);
Px3= Padre3(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Pad3=Px3(L,:);
end
X32=MaxTard3==MinTard3;
 if MaxTard3==MinTard3;
   MinDifti3=min(MatTsP31);
   k= find(MinDifti3==MatTsP31);
Px3= Padre3(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
Pad3=Px3(L,:);
 end
 X33=MaxTard3~=MinTard3;
 X34=MaxDti3~=MinDti3;
if MaxTard3~=MinTard3;
   if MaxDti3~=MinDti3;
for i=1:n;
   j=1;
% Normalizamos funciones de Makespan  y Tardanza:
fDifti3(i,j)=(MaxDti3- MatTsP31N(i,1))/(MaxDti3- MinDti3);
fTmax3(i,j)=(MaxTard3- MatTsP32N(i,1))/(MaxTard3- MinTard3);
U3(i,j)= W1*fDifti3(i,1)+ W2*fTmax3(i,1);
end
if searchx>0;
   U3(searchx)=0;
   Up3=U3;
else
   Up3=U3;
end
MaxU3=max(Up3);
k= find(MaxU3==Up3);
Px3= Padre3(k,:);
K= size(k);
```

```
l= K(1,1);
L= randi(l);
Pad3=Px3(L,:);
   end
end
LT= [k(L,1)];
Ltb5=Perm3(LT,:);
Ltb6=[Ltb5(1,2), Ltb5(1,1)];
Listatabu3=[Listatabu2; Ltb5; Ltb6];
%%%%
for w=1:Wts;
   Padrepp=Pad3;
   Ltbf=Listatabu3;
for i=1:n;
   j=1:SB;
  tx=randperm(SB,2);%Posiciones en secini a cruzar
   B1=Pad2(1,tx(1,1));
   B2=Pad2(1,tx(1,2));
   t= [B1 B2];
    if t(1,1)==Ltbf(1,1)&t(1,2)==Ltbf(1,2);
       t(1,:)=[0];
    elseif t(1,1)==Ltbf(2,1)&t(1,2)==Ltbf(2,2);
t(1,:)=[0];
 elseif t(1,1)==Ltbf(3,1)&t(1,2)==Ltbf(3,2);
t(1,:)=[0];
elseif t(1,1)==Ltbf(4,1)&t(1,2)==Ltbf(4,2);
t(1,:)=[0];
elseif t(1,1)==Ltbf(5,1)&t(1,2)==Ltbf(5,2);
t(1,:)=[0];
elseif t(1,1)==Ltbf(6,1)&t(1,2)==Ltbf(6,2);
t(1,:)=[0];
    end
     BB1=find(t(1,1)==Padrepp);
   BB2=find(t(1,2)==Padrepp);
   Padrepp(1,BB1)=t(1,2);
   Padrepp(1,BB2)=t(1,1);
    Pad4(i,j)=Padrepp(:,j);

 for j= 1:2;
      Perm4(i,j)= t(:,j);
 end
end
for i= 1:n;
TsPW= RKQFFSfitness (PT,N,ME,Dd,Bd,yk,SecProg,Pad4(i,:),CjSec,CjNi,RT);
```

```
for j=1:3;
   MatTsPW(i,j)= TsPW(:,j);
end
end
 MatTsPW1=MatTsPW(:,1);
 MatTsPW2=MatTsPW(:,2);
 MatTsPW1N=MatTsPW1;
 MatTsPW2N=MatTsPW2;
%%%
searchx=find(Perm4(:,1)==0);
if searchx>0;
 MatTsPW1(searchx)=0;
 MaxDtiW= max(MatTsPW1);
 MatTsPW1(searchx)=A;
 MinDtiW= min(MatTsPW1);
 MatTsPW2(searchx)=0;
 MaxTardW=max(MatTsPW2);
 MatTsPW2(searchx)=A;
 MinTardW=min(MatTsPW2);
else
MaxDtiW= max(MatTsPW1);
MinDtiW= min(MatTsPW1);
MaxTardW=max(MatTsPW2);
MinTardW=min(MatTsPW2);
end
XW1=MaxDtiW==MinDtiW;
if MaxDtiW==MinDtiW;
   MinTardanzaW=min(MatTsPW2);
   k= find(MinTardanzaW==MatTsPW2);
PxW= Pad4(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
PadreF=PxW(L,:);
Pad3=PadreF;
end
XW2=MaxTardW==MinTardW;
 if MaxTardW==MinTardW;
   MinDiftiW=min(MatTsPW1);
   k= find(MinDiftiW==MatTsPW1);
PxW= Pad4(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
```

```
PadreF=PxW(L,:);
Pad3=PadreF;
 end
 XW3=MaxTardW~=MinTardW;
 XW4=MaxDtiW~=MinDtiW;
if MaxTardW~=MinTardW;
   if MaxDtiW~=MinDtiW;
for i=1:n;
   j=1;
% Normalizamos funciones de Makespan  y Tardanza:
fDiftiW(i,j)=(MaxDtiW- MatTsPW1N(i,1))/(MaxDtiW- MinDtiW);
fTmaxW(i,j)=(MaxTardW- MatTsPW2N(i,1))/(MaxTardW- MinTardW);
UW(i,j)= W1*fDiftiW(i,1)+ W2*fTmaxW(i,1);
end
if searchx>0;
   UW(searchx)=0;
   UpW=UW;
else
   UpW=UW;
end
MaxUW=max(UpW);
k= find(MaxUW==UpW);
PxW= Pad4(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
PadreF=PxW(L,:);
Pad3=PadreF;
   end
end
LT= [k(L,1)];
Ltb7=Perm4(LT,:);
Ltb8=[Ltb7(1,2), Ltb7(1,1)];
Listatabu3(1,:)=[Ltbf(3,1),Ltbf(3,2)];
Listatabu3(2,:)=[Ltbf(4,1),Ltbf(4,2)];
Listatabu3(3,:)=[Ltbf(5,1),Ltbf(5,2)];
Listatabu3(4,:)=[Ltbf(6,1),Ltbf(6,2)];
Listatabu3(5,:)=[Ltb7(1,1), Ltb7(1,2)];
Listatabu3(6,:)=[Ltb8(1,1), Ltb8(1,2)];
j= 1:SB;
Padres(w,j)=PadreF(:,j);
j=1:3;
TsPFF(w,j)= RKQFFSfitness (PT,N,ME,Dd,Bd,yk,SecProg,Padres(w,:),CjSec,CjNi,RT);
end
```

```
MaxCTPF= max(TsPFF);
MinCTPF= min(TsPFF);
if MaxCTPF(1,2)==MinCTPF(1,2);
   j=1;
   DifTiPF=TsPFF(:,j);
   MinDifTiPF=min(DifTiPF);
   k= find(MinDifTiPF==DifTiPF);
PF= Padres(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
PadreFF=PF(L,:);
end
if MaxCTPF(1,1)==MinCTPF(1,1);
   j=2;
 TmaxxPF=TsPFF(:,j);
   MinTmaxxPF=min(TmaxxPF);
   k= find(MinTmaxxPF==TmaxxPF);
PF= Padres(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
PadreFF=PF(L,:);
end
if MaxCTPF(1,2)~=MinCTPF(1,2);
if MaxCTPF(1,1)~=MinCTPF(1,1);
for i=1:w;
   j=1;
% Normalizamos funciones de Makespan  y Tardanza:
fDifTiPF(i,j)=(MaxCTPF(1,1)- TsPFF(i,1))/(MaxCTPF(1,1)- MinCTPF(1,1));
fTmaxPF(i,j)=(MaxCTPF(1,2)- TsPFF(i,2))/(MaxCTPF(1,2)- MinCTPF(1,2));
%Con la ecuación siguiente se determina cual de las secuencias da un
%mejor resultado. Se colocó un peso w del 0.4 para Makespan  y 0.6 para
%tardanza. La solución cuyo resultado sea mayor acercandose a uno, es la
%elegida como mejor secuencia.
UFF(i,j)= W1*fDifTiPF(i,1)+ W2*fTmaxPF(i,1);
end
MaxUFF=max(UFF);
k= find(MaxUFF==UFF);
PF= Padres(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
PadreFF=PF(L,:);
```

```
end
end
PadreTS=PadreFF;
end
```

Crossover

```
function Desc= RCrossover(parent1,parent2,PT,N,ME,Dd,Bd,yk,SecProg,CjSec,CjNi, W1, W2,RT)
%Función para generar dos hijos, luego del cruzammiento de sus padres.
 [SA SB]= size(parent1);
A=2000000;
B=1000000;
if SB==2;
perm=1;
y=2;
else
N2=SB/2-1;
N3= round(N2);
perm=randperm(SB,1);
x=perm+randperm(N3,1);
if x>SB;
   y=SB;
else
   y=x;
end
end
Cross= (perm:y);
Cross1=parent1(Cross);
Cross2=parent2(Cross);
parent1(perm:y)=Cross2;
parent2(perm:y)=Cross1;
parent1(perm:y)=A;
parent2(perm:y)=A;
l=size(Cross);
l1=l(1,2);
for i=1:l1;
u= find(Cross2(1,i)==parent1);
if u>0;
   u1=u;
else
   u1=0;
end
j=1;
```

```
u2(i,j)= u1(:,j);
end
for i=1:l1;
   if u2(i,1)>0;
      parent1(1,u2(i,1))=B;
   end
end
parent1(perm:y)= Cross2;
sizeCross= size(Cross);
TamCross= sizeCross(1,2);
for i=1:TamCross;
   fcross= find(Cross1(1,i)==parent1);
   if fcross>0;
      rep= fcross;
   else
      rep=0;
   end
   j=1;
   repet(i,j)= rep(:,j);
end
for i=1:TamCross;
   if repet(i,1)==0;
      fcross1=find(B==parent1);
      pos1fcross1=fcross1(1,1);
      parent1(1,pos1fcross1)=Cross1(1,i);
   end
end
%%%Para parent2
for i=1:l1;
up2= find(Cross1(1,i)==parent2);
if up2>0;
   u1p2=up2;
else
   u1p2=0;
end
j=1;
u2p2(i,j)= u1p2(:,j);
end
for i=1:l1;
   if u2p2(i,1)>0;
      parent2(1,u2p2(i,1))=B;
   end
end
parent2(perm:y)= Cross1;
```

```
sizeCrossp2= size(Cross);
TamCrossp2= sizeCrossp2(1,2);
for i=1:TamCrossp2;
   fcrossp2= find(Cross2(1,i)==parent2);
   if fcrossp2>0;
      repp2= fcrossp2;
   else
      repp2=0;
   end
   j=1;
   repetp2(i,j)= repp2(:,j);
end
for i=1:TamCrossp2;
   if repetp2(i,1)==0;
      fcross1p2=find(B==parent2);
      pos1fcross1p2=fcross1p2(1,1);
      parent2(1,pos1fcross1p2)=Cross2(1,i);
   end
end
Hijos= [parent1; parent2];
for i= 1:2;
 DifTCross(i,:)= RKQFFSfitness (PT,N,ME,Dd,Bd,yk,SecProg,Hijos(i,:),CjSec,CjNi,RT);
end
% MinTiCros= min(DifTiCros);
%     k= find(MinTiCros==DifTiCros);
% PxC= Hijos(k,:);
% K= size(k);
% l= K(1,1);
% L= randi(l);
% Desc=PxC(L,:);
%%%%%%%%%%%%%%%%
MaxCTH= max(DifTCross);
MinCTH= min(DifTCross);
if DifTCross(1,1)==DifTCross(2,1);
   if  DifTCross(1,2)>DifTCross(2,2);
      Hijo= Hijos(2,:);
      DifTiHijo= DifTCross(2,:);
   else
       Hijo= Hijos(1,:);
       DifTiHijo= DifTCross(1,:);
   end
else
if DifTCross(1,2)==DifTCross(2,2);
   if  DifTCross(1,1)>DifTCross(2,1);
```

```
      Hijo= Hijos(2,:);
      DifTiHijo= DifTCross(2,:);
   else
       Hijo= Hijos(1,:);
       DifTiHijo= DifTCross(1,:);
   end
end
end
if DifTCross(1,1)~=DifTCross(2,1);
   if DifTCross(1,2)~=DifTCross(2,2);
% else
for i=1:2;
   j=1;
fDifTiH(i,j)=(MaxCTH(1,1)- DifTCross(i,1))/(MaxCTH(1,1)- MinCTH(1,1));
fTmaxH(i,j)=(MaxCTH(1,2)- DifTCross(i,2))/(MaxCTH(1,2)- MinCTH(1,2));
UH(i,j)= W1*fDifTiH(i,1)+ W2*fTmaxH(i,1);
end
MaxUH=max(UH);
k= find(MaxUH==UH);
K= size(k);
l= K(1,1);
if l>1;
L= randi(l);
Hijo= Hijos(L,:);

else
   Hijo=Hijos(k,:);
end
   end
end
Desc= Hijo;
end
```

Mutación

```
function Mut= RMutacion (PT,N,ME,Dd,Bd,yk,SecProg,Hijo,CjSec,CjNi, W1, W2,RT)
%Con esta función se somete a mutación el hijo obtenido del cruzamiento,
%intercambiando dos posciones dentro de la secuencia hijo. Además evalua
%tanto al hijo como a la mutación y escoge cual de los dos es la mejor
%solución
HijoM= [Hijo];
[SA SB]= size(HijoM);
if SB==2;
perm=1;
```

```
y=2;
else
N2=SB/2-1;
N3= round(N2);
perm=randperm(N3,1);
x=perm+randperm(N3,1);
if x>SB;
   y=SB;
else
   y=x;
end
end
posmix= (perm:y);
comb= HijoM(perm:y);
l= size(comb);
L=l(1,2);
mix=randperm(L);
comb2=comb(:,mix);
HijoM(perm:y)=comb2;
Hijos=[Hijo; HijoM];
for i=1:2;
DifTiM(i,:)= RKQFFSfitness (PT,N,ME,Dd,Bd,yk,SecProg,Hijos(i,:),CjSec,CjNi,RT);
end
MaxCTH= max(DifTiM);
MinCTH= min(DifTiM);
if DifTiM(1,1)==DifTiM(2,1);
   if  DifTiM(1,2)>DifTiM(2,2);
      Hijo= Hijos(2,:);
      DifTiHijo= DifTiM(2,:);
   else
       Hijo= Hijos(1,:);
       DifTiHijo= DifTiM(1,:);
   end
else
if DifTiM(1,2)==DifTiM(2,2);
   if  DifTiM(1,1)>DifTiM(2,1);
      Hijo= Hijos(2,:);
      DifTiHijo= DifTiM(2,:);
   else
       Hijo= Hijos(1,:);
       DifTiHijo= DifTiM(1,:);
   end
end
end
```

```
if DifTiM(1,1)~=DifTiM(2,1);
   if DifTiM(1,2)~=DifTiM(2,2);
% else
for i=1:2;
   j=1;
fDifTiM(i,j)=(MaxCTH(1,1)- DifTiM(i,1))/(MaxCTH(1,1)- MinCTH(1,1));
fTmaxH(i,j)=(MaxCTH(1,2)- DifTiM(i,2))/(MaxCTH(1,2)- MinCTH(1,2));
UH(i,j)= W1*fDifTiM(i,1)+ W2*fTmaxH(i,1);
end
MaxUH=max(UH);
k= find(MaxUH==UH);
K= size(k);
l= K(1,1);
if l>1;
L= randi(l);
Hijo= Hijos(L,:);

else
   Hijo=Hijos(k,:);
end
   end
end
Mut=Hijo;
% MinTimut= min(DifTiM);
%    k= find(MinTimut==DifTiM);
% PxC= Hijos(k,:);
% K= size(k);
% l= K(1,1);
% L= randi(l);
% Mut=PxC(L,:);
end

%%%
% Mut= Hijo;
```

2 Opt

```
function [BestSol]= opt2(PT,N,ME,Dd,Bd,yk,SecProg,Besthijo,CjSec,CjNi,W1,W2,RT)
NSecProg=SecProg;
P=Besthijo;
P2=P;
[A B]=size(ME);
[SA SB]=size(P);
Sum=0;
X=PT(NSecProg,:);
```

```
Dds= Dd(NSecProg,:);
tx1=randperm(SB-1,1);
tx2= tx1+1;
Px1= P(1,tx2);
Px2= P(1,tx1);
P2(1,tx1)=Px1;
P2(1,tx2)=Px2;
MP=[P;P2];
for i=1:2;
DiftiTar(i,:)= RKQFFSfitness (PT,N,ME,Dd,Bd,yk,SecProg,MP(i,:),CjSec,CjNi,RT);
end
MaxCTH= max(DiftiTar);
MinCTH= min(DiftiTar);
if DiftiTar(1,1)==DiftiTar(2,1);
   if  DiftiTar(1,2)>DiftiTar(2,2);
       Popt= MP(2,:);
       DifTiPopt= DiftiTar(2,:);
   else
        Popt= MP(1,:);
        DifTiPopt= DiftiTar(1,:);
   end
else
if DiftiTar(1,2)==DiftiTar(2,2);
   if  DiftiTar(1,1)>DiftiTar(2,1);
       Popt= MP(2,:);
       DifTiPopt= DiftiTar(2,:);
   else
        Popt= MP(1,:);
        DifTiPopt= DiftiTar(1,:);
   end
end
end
if DiftiTar(1,1)~=DiftiTar(2,1);
   if DiftiTar(1,2)~=DiftiTar(2,2);
% else
for i=1:2;
   j=1;
fDiftiTar(i,j)=(MaxCTH(1,1)- DiftiTar(i,1))/(MaxCTH(1,1)- MinCTH(1,1));
fTmaxH(i,j)=(MaxCTH(1,2)- DiftiTar(i,2))/(MaxCTH(1,2)- MinCTH(1,2));
UH(i,j)= W1*fDiftiTar(i,1)+ W2*fTmaxH(i,1);
end
MaxUH=max(UH);
k= find(MaxUH==UH);
K= size(k);
```

```
l= K(1,1);
if l>1;
L= randi(l);
Popt= MP(L,:);

else
   Popt=MP(k,:);
end
   end
end
BestSol=Popt;
end
```

Algoritmo genetico- Cuerpo principal

```
clc, clear all;
%La población inicial se realiza de manera aleatoria
%para elegir los padres
TP= 'AGChuBRep';
PT= xlsread(TP,3);%Tiempo de procesamiento de cada trabajo
%en cada máquina
S= size(PT); %Tamaño de la matriz
N= S(1,1);%trabajos
E= S(1,2);%Estaciones
ME=[2 1 5 4 3];%# de máquinas por estaciones
Dd= xlsread(TP,7);% Tiempo maximo de entrega por trabajo
SecProg= [9    4   12   18   10   16    7   14    5    1    6   20    3   17    8   11    2   15   19
13];
CmaxTmaxSecProg=[1288    0];
[A B]=size(ME);
WAg=10;
Wts=20;
W1=0.5;
W2=0.5;
B1=0; %Bd de reprogramacion anterior en su defecto colocar igual a cero
CjSec=xlsread(TP,8);
Xj=PT(SecProg,:);
RT= xlsread(TP,9);
ME1f=ME(1,1);
Sum=0;
 for i=1:B;
      Sum=Sum+ME(1,i);
 end
```

```
 MENi= Sum-ME(1,B)+1;
 MENf=Sum;
%%%Tiempo de inicio para cada trabajo obtenido de CjSec
for i=2:N;
   j=1:ME1f;
   XCjsec=CjSec(i,j)~=CjSec(i-1,j);
   FCjini= find(XCjsec==1);
   Cjini=CjSec(i,FCjini)-Xj(i,1);
j=1;
CjNi(i,j)= Cjini(:,j)
end

Bxx= B1-1;
if Bxx<0;
Bxx=0;
end
B2= N-Bxx;
AleB2= randperm(B2,1);
Bd= Bxx+AleB2
% Bd=11
Bx= SecProg(1,Bd)
if Bd==1;
Sec2=[SecProg(1:N)];
else
   Sec1=[SecProg(1:Bd-1)];
   Sec2=[SecProg(Bd:N)];
   end
Bdt= round(CmaxTmaxSecProg(1,1)- CjNi(Bd,1));
yyk=randi(Bdt,1);
yk= CjNi(Bd,1) + yyk
[S1 S2]=size(Sec2);
NsecProg=SecProg(1:N);
Padini1= [SecProg(Bd:N)];
pxx= Bd+1;
Padini2= [SecProg(pxx:N),Padini1(1,1)];
for i=1:WAg;
% Utilizamos busqueda tabu para buscar el Padre #1 con
% secuencia inicial=Padini1, de igual forma para el Padre #2
% usando como secuencia inicial=Padini2;
Parent1= RTsearchFFS(Padini1,PT,N,ME,Dd,Bd,yk,SecProg,CjSec,CjNi,Wts, W1, W2,RT);
Parent2= RTsearchFFS(Padini2,PT,N,ME,Dd,Bd,yk,SecProg,CjSec,CjNi,Wts, W1, W2,RT);
BestHijoCruce= RCrossover(Parent1,Parent2,PT,N,ME,Dd,Bd,yk,SecProg,CjSec,CjNi, W1, W2,RT);
BestHijoFinal= BestHijoCruce;
%%%% Mutación
```

```
pm= rand(1);
 if pm<0.4;
BestHijoFinal= RMutacion(PT,N,ME,Dd,Bd,yk,SecProg,BestHijoFinal,CjSec,CjNi, W1, W2,RT);
 end
 BestHijoSol= opt2(PT,N,ME,Dd,Bd,yk,SecProg,BestHijoFinal,CjSec,CjNi,W1,W2,RT);
%%
for j=1:S2;
MtBesthijo(i,j)=BestHijoSol(:,j);
end
end
for i=1:WAg;
   j=1:3;
TptiBH(i,j)= RKQFFSfitness (PT,N,ME,Dd,Bd,yk,SecProg,MtBesthijo(i,:),CjSec,CjNi,RT);
end
MaxCTF= max(TptiBH);
MinCTF= min(TptiBH);
if MaxCTF(1,2)==MinCTF(1,2);
 j=1;
   DifTiPF=TptiBH(:,j);
   MinDifTiPF=min(DifTiPF);
   k= find(MinDifTiPF==DifTiPF);
Px2= MtBesthijo(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
SolFAG=Px2(L,:);
end
if MaxCTF(1,1)==MinCTF(1,1);
   j=2;
   TmaxPF=TptiBH(:,j);
   MinTmaxPF=min(TmaxPF);
   k= find(MinTmaxPF==TmaxPF);
Px2= MtBesthijo(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
SolFAG=Px2(L,:);
end
if MaxCTF(1,2)~=MinCTF(1,2);
if MaxCTF(1,1)~=MinCTF(1,1);
for i=1:WAg;
   j=1;
fDifPF(i,j)=(MaxCTF(1,1)- TptiBH(i,1))/(MaxCTF(1,1)- MinCTF(1,1));
fTmaxPF(i,j)=(MaxCTF(1,2)- TptiBH(i,2))/(MaxCTF(1,2)- MinCTF(1,2));
```

```
UPF(i,j)= W1*fDifPF(i,1)+ W2*fTmaxPF(i,1);
end
MaxUPF=max(UPF);
k= find(MaxUPF==UPF);
Px2= MtBesthijo(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
SolFAG=Px2(L,:);
end
end
MatrizF=TptiBH;
SolSecFinalAG=SolFAG
SolSecFTiTard=RKQFFSfitness (PT,N,ME,Dd,Bd,yk,SecProg,SolSecFinalAG,CjSec,CjNi,RT)
RT(Bx,1)=yk
NSecProg=SecProg;
NSecProg(Bd:N)=SolSecFinalAG;
SolF='SecFin';
ExcSF=xlswrite(SolF,NSecProg,1)
Reltm= RT(NSecProg,:);%% Release time ordenado segun nueva secuencia
%%%% Calculo de Cj para la nueva secuencia
NewRT= 'RTN';
ExcRT=xlswrite(NewRT,RT,1);
CJRP= KqCj (PT,SecProg,SolSecFinalAG,N,Bd,ME,Dd,CjNi,CjSec,Reltm);
CJF= 'CJFreprog';
ExcCJ=xlswrite(CJF,CJRP,1);
```

4. Modificaciones modelo de reprogramación

Función normalizer

```
function [BestSec]= RNormalizar (MtBesthijo,PT,N,ME,Dd,Bd,yk,SecProg,P,CjSec,CjNi,RT,
W1,W2)
[A B]= size(P);
MaxCTF= max(P);
MinCTF= min(P);
if MaxCTF(1,2)==MinCTF(1,2);
 j=1;
   DifTiPF=P(:,j);
   MinDifTiPF=min(DifTiPF);
   k= find(MinDifTiPF==DifTiPF);
Px2= MtBesthijo(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
```

```
SolFAG=Px2(L,:);
end
if MaxCTF(1,1)==MinCTF(1,1);
   j=2;
   TmaxPF=P(:,j);
   MinTmaxPF=min(TmaxPF);
   k= find(MinTmaxPF==TmaxPF);
Px2= MtBesthijo(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
SolFAG=Px2(L,:);
end
if MaxCTF(1,2)~=MinCTF(1,2);
if MaxCTF(1,1)~=MinCTF(1,1);
for i=1:A;
   j=1;
fDifPF(i,j)=(MaxCTF(1,1)- P(i,1))/(MaxCTF(1,1)- MinCTF(1,1));
fTmaxPF(i,j)=(MaxCTF(1,2)- P(i,2))/(MaxCTF(1,2)- MinCTF(1,2));
UPF(i,j)= W1*fDifPF(i,1)+ W2*fTmaxPF(i,1);
end
MaxUPF=max(UPF);
k= find(MaxUPF==UPF);
Px2= MtBesthijo(k,:);
K= size(k);
l= K(1,1);
L= randi(l);
SolFAG=Px2(L,:);
end
end
BestSec=SolFAG;
end
```

Algoritmo genetico – Cuerpo principal

```
PT= xlsread(TP,1);%Tiempo de procesamiento de cada trabajo
%en cada máquina
S= size(PT); %Tamaño de la matriz
N= S(1,1);%trabajos
E= S(1,2);%Estaciones
ME=[3 1 5 3];%# de máquinas por estaciones
Dd= xlsread(TP,2);% Tiempo maximo de entrega por trabajo
SecProg= [1 2 3 4 5];
```

```
CmaxTmaxSecProg=[2782  49726];
[A B]=size(ME);
Wpi=20;
WAg=100;
X=2;
Wts=20;
W1=0.9;
W2=0.1;
B1=251; %Bd de reprogramacion anterior, en su defecto colocar igual a cero
CjSec=xlsread(TP,3);
Xj=PT(SecProg,:);
RT= xlsread(TP,4);
ME1f=ME(1,1);
Sum=0;
 for i=1:B;
      Sum=Sum+ME(1,i);
 end
 MENi= Sum-ME(1,B)+1;
 MENf=Sum;
%%%Tiempo de inicio para cada trabajo obtenido de CjSec
for i=2:N;
   j=1:ME1f;
   XCjsec=CjSec(i,j)~=CjSec(i-1,j);
   FCjini= find(XCjsec==1);
   Cjini=CjSec(i,FCjini)-Xj(i,1);
j=1;
CjNi(i,j)= Cjini(:,j);
end

Bxx= B1-1;
if Bxx<0;
Bxx=0;
end
B2= N-Bxx;
AleB2= randperm(B2,1);
% Bd= Bxx+AleB2
Bd=311
Bx= SecProg(1,Bd)
if Bd==1;
Sec2=[SecProg(1:N)];
else
   Sec1=[SecProg(1:Bd-1)];
   Sec2=[SecProg(Bd:N)];
   end
```

```
Bdt= round(CmaxTmaxSecProg(1,1)- CjNi(Bd,1));
yyk=randi(Bdt,1);
% yk= CjNi(Bd,1) + yyk
yk=2496.5
[S1 S2]=size(Sec2);
NsecProg=SecProg(1:N);
Padini1= [SecProg(Bd:N)];
pxx= Bd+1;
Padini2= [SecProg(pxx:N),Padini1(1,1)];

for i=1:Wpi;
% Utilizamos busqueda tabu para buscar el Padre #1 con
% secuencia inicial=Padini1, de igual forma para el Padre #2
% usando como secuencia inicial=Padini2;
Parent1= RTsearchFFS(Padini1,PT,N,ME,Dd,Bd,yk,SecProg,CjSec,CjNi,Wts, W1, W2,RT);
Parent2= RTsearchFFS(Padini2,PT,N,ME,Dd,Bd,yk,SecProg,CjSec,CjNi,Wts, W1, W2,RT);
BestHijoCruce1=
RCrossover(Parent1,Parent2,PT,N,ME,Dd,Bd,yk,SecProg,CjSec,CjNi,W1,W2,RT);
BestHijoFinal1= BestHijoCruce1;
%%%% Mutación
pm= rand(1);
 if pm<0.4;
BestHijoFinal1= RMutacion(PT,N,ME,Dd,Bd,yk,SecProg,BestHijoFinal1,CjSec,CjNi,W1,W2,RT);
 end
 BestHijoSol1= opt2(PT,N,ME,Dd,Bd,yk,SecProg,BestHijoFinal1,CjSec,CjNi,W1,W2,RT);
%%%
j=1:S2;
MtBesthijo1(i,j)=BestHijoSol1(:,j);
%%%
j=1:3;
TptiBH1(i,j)= RKQFFSfitness(PT,N,ME,Dd,Bd,yk,SecProg,MtBesthijo1(i,:),CjSec,CjNi,RT);
end
SolFAG1= RNormalizar(MtBesthijo1,PT,N,ME,Dd,Bd,yk,SecProg,TptiBH1,CjSec,CjNi,RT, W1, W2);
%%%%%
for i= 1:WAg;
   % Utilizamos busqueda tabu para buscar el Padre #1 con
% secuencia inicial=Padini1, de igual forma para el Padre #2
% usando como secuencia inicial=Padini2;
Parent1= RTsearchFFS(Padini1,PT,N,ME,Dd,Bd,yk,SecProg,CjSec,CjNi,Wts, W1, W2,RT);
Parent2= RTsearchFFS(Padini2,PT,N,ME,Dd,Bd,yk,SecProg,CjSec,CjNi,Wts, W1, W2,RT);
BestHijoCruce2=
RCrossover(Parent1,Parent2,PT,N,ME,Dd,Bd,yk,SecProg,CjSec,CjNi,W1,W2,RT);
BestHijoFinal2= BestHijoCruce2;
%%%% Mutación
```

```
pm= rand(1);
 if pm<0.4;
BestHijoFinal2= RMutacion(PT,N,ME,Dd,Bd,yk,SecProg,BestHijoFinal2,CjSec,CjNi,W1,W2,RT);
 end
 BestHijoSol2= opt2(PT,N,ME,Dd,Bd,yk,SecProg,BestHijoFinal2,CjSec,CjNi,W1,W2,RT);
%%%
j=1:S2;
MtBesthijo2(i,j)=BestHijoSol2(:,j);
%%%
j=1:3;
TptiBH2(i,j)= RKQFFSfitness(PT,N,ME,Dd,Bd,yk,SecProg,MtBesthijo2(i,:),CjSec,CjNi,RT);
%  Xk=WAg/2;
 if i== WAg/X;
%     break
    SolFAG2= RNormalizar(MtBesthijo2,PT,N,ME,Dd,Bd,yk,SecProg,TptiBH2,CjSec,CjNi,RT, W1,
W2);
    SolFAX1=[SolFAG1; SolFAG2];
    TptiBHAG1= RKQFFSfitness(PT,N,ME,Dd,Bd,yk,SecProg,SolFAG1,CjSec,CjNi,RT);
    TptiBHAG2= RKQFFSfitness(PT,N,ME,Dd,Bd,yk,SecProg,SolFAG2,CjSec,CjNi,RT);
    TptiBHX1=[TptiBHAG1; TptiBHAG2];
    NormSolAG= RNormalizar(SolFAX1,PT,N,ME,Dd,Bd,yk,SecProg,TptiBHX1,CjSec,CjNi,RT, W1,
W2);
    if NormSolAG==SolFAG2
       break
    end
%     NormSolFAX1=
 end
end
     SolFAG2= RNormalizar(MtBesthijo2,PT,N,ME,Dd,Bd,yk,SecProg,TptiBH2,CjSec,CjNi,RT, W1,
W2);
    SolFAX1=[SolFAG1; SolFAG2];
    TptiBHAG1= RKQFFSfitness(PT,N,ME,Dd,Bd,yk,SecProg,SolFAG1,CjSec,CjNi,RT);
    TptiBHAG2= RKQFFSfitness(PT,N,ME,Dd,Bd,yk,SecProg,SolFAG2,CjSec,CjNi,RT);
    TptiBHX1=[TptiBHAG1; TptiBHAG2];
    NormSolAG= RNormalizar(SolFAX1,PT,N,ME,Dd,Bd,yk,SecProg,TptiBHX1,CjSec,CjNi,RT, W1,
W2)
    %%%%
% SolSecFinalAG=SolFAG
SolSecFTiTard=RKQFFSfitness (PT,N,ME,Dd,Bd,yk,SecProg,NormSolAG,CjSec,CjNi,RT)
RT(Bx,1)=yk
NSecProg=SecProg;
NSecProg(Bd:N)=NormSolAG;
SolF='SecFin';
ExcSF=xlswrite(SolF,NSecProg,1)
```

```
Reltm= RT(NSecProg,:);%% Release time ordenado segun nueva secuencia
%%%% Calculo de Cj para la nueva secuencia
NewRT= 'RTN';
ExcRT=xlswrite(NewRT,RT,1);
CJRP= KqCj (PT,SecProg,NormSolAG,N,Bd,ME,Dd,CjNi,CjSec,Reltm);
CJF= 'CJFreprog';
ExcCJ=xlswrite(CJF,CJRP,1);

% end
```

5. Tiempos de proceso caso de estudio de fábrica de calzado

Lote	Etapa 1	Etapa 2	Etapa 3	Etapa 4	Due date
1	14,22	4,374	23,94	32,22	1350
2	14,22	4,374	23,94	32,22	1350
3	17,38	5,346	29,26	39,38	1350
4	15,01	4,617	25,27	34,01	1350
5	16,59	5,103	27,93	37,59	1350
6	17,8	4,86	25,25	10,8	540
7	16,02	4,374	22,725	9,72	540
8	16,02	4,374	22,725	9,72	540
9	18,69	5,103	26,5125	11,34	540
10	19,58	5,346	27,775	11,88	540
11	17,8	4,86	25,25	10,8	540
12	19,58	5,346	27,775	11,88	540
13	16,02	4,374	22,725	9,72	540
14	18,69	5,103	26,5125	11,34	540
15	19,58	5,346	27,775	11,88	540
16	18,69	5,103	26,5125	11,34	540
17	16,91	4,617	23,9875	10,26	540
18	17,1	5,13	22,5625	9,88	1350
19	18,9	5,67	24,9375	10,92	1350
20	16,2	4,86	21,375	9,36	1350
21	18	5,4	23,75	10,4	1350
22	15,66	4,86	21,24	34,2	540
23	19,14	5,94	25,96	41,8	540
24	17,4	5,4	23,6	38	540
25	18,27	5,67	24,78	39,9	540

26	15,66	4,86	21,24	34,2	540
27	19,14	5,94	25,96	41,8	540
28	16,53	5,13	22,42	36,1	540
29	15,66	4,86	21,24	34,2	540
30	19,14	5,94	25,96	41,8	540
31	19,14	5,94	25,96	41,8	540
32	19,14	5,94	25,96	41,8	540
33	16,53	5,13	22,42	36,1	540
34	16,53	5,13	22,42	36,1	540
35	16,53	5,13	22,42	36,1	540
36	19,14	5,94	25,96	41,8	540
37	16,17	5,67	26,25	10,92	1350
38	16,17	5,67	26,25	10,92	1350
39	16,17	5,67	26,25	10,92	1350
40	16,17	5,67	26,25	10,92	1350
41	13,86	4,86	22,5	9,36	1350
42	16,15	5,13	23,94	9,12	1080
43	16,15	5,13	23,94	9,12	1080
44	16,15	5,13	23,94	9,12	1080
45	17	5,4	25,2	9,6	1080
46	17,85	5,67	26,46	10,08	1080
47	16,15	5,13	23,94	9,12	1080
48	15,3	4,86	22,68	8,64	1080
49	16,15	5,13	23,94	9,12	1080
50	16,15	5,13	23,94	9,12	1080
51	17,85	5,67	26,46	10,08	1080
52	17	5,4	25,2	9,6	1080
53	18,27	5,67	26,565	31,92	1080
54	18,27	5,67	26,565	31,92	1080
55	15,66	4,86	22,77	27,36	1080
56	19,14	5,94	27,83	33,44	1080
57	15,66	4,86	22,77	27,36	1080
58	19,14	5,94	27,83	33,44	1080
59	17,4	5,4	25,3	30,4	1080
60	18,27	5,67	26,565	31,92	1080
61	16,53	5,13	24,035	28,88	1080
62	15,66	4,86	22,77	27,36	1080
63	19,2	5,4	30,6	10,8	1080
64	19,2	5,4	30,6	10,8	1080
65	17,28	4,86	27,54	9,72	1080
66	18,24	5,13	29,07	10,26	1080

67	19,2	5,4	30,6	10,8	1080
68	16,3065	5,67	30,45	10,5	3000
69	14,7535	5,13	27,55	9,5	3000
70	13,977	4,86	26,1	9	3000
71	16,3065	5,67	30,45	10,5	3000
72	17,083	5,94	31,9	11	3000
73	17,083	5,94	31,9	11	3000
74	14,7535	5,13	27,55	9,5	3000
75	15,53	5,4	29	10	3000
76	17,083	5,94	31,9	11	3000
77	14,7535	5,13	27,55	9,5	3000
78	13,977	4,86	26,1	9	3000
79	14,7535	5,13	27,55	9,5	3000
80	16,3065	5,67	30,45	10,5	3000
81	17,005	5,643	26,79	9,88	3000
82	19,69	6,534	31,02	11,44	3000
83	17,9	5,94	28,2	10,4	3000
84	18,795	6,237	29,61	10,92	3000
85	17,005	5,643	26,79	9,88	3000
86	17,9	5,94	28,2	10,4	3000
87	19,69	6,534	31,02	11,44	3000
88	16,11	5,346	25,38	9,36	3000
89	17,005	5,643	26,79	9,88	3000
90	19,69	6,534	31,02	11,44	3000
91	17,6	6,534	25	35,2	1350
92	17,6	6,534	34	35,2	1350
93	16	5,94	25	32	1350
94	14,4	5,346	30	28,8	1350
95	16,8	6,237	22	33,6	1350
96	19,593	6,237	32,403	10,5	2000
97	19,593	6,237	32,403	10,5	2000
98	16,794	5,346	27,774	9	2000
99	19,593	6,237	32,403	10,5	2000
100	19,593	6,237	32,403	10,5	2000
101	16,794	5,346	27,774	9	2000
102	18,66	5,94	30,86	10	2000
103	19,593	6,237	32,403	10,5	2000
104	17,727	5,643	29,317	9,5	2000
105	20,526	6,534	33,946	11	2000
106	19,593	6,237	32,403	10,5	2000
107	18,66	5,94	30,86	10	2000

108	20,526	6,534	33,946	11	2000
109	18,66	5,94	30,86	10	2000
110	19,593	6,237	32,403	10,5	2000
111	20,118	6,237	30,45	11,34	1350
112	19,16	5,94	29	10,8	1350
113	18,202	5,643	27,55	10,26	1350
114	18,202	5,643	27,55	10,26	1350
115	18,202	5,643	27,55	10,26	1350
116	20,118	6,237	30,45	11,34	1350
117	13,689	5,346	26,739	9,72	2500
118	15,9705	6,237	31,1955	11,34	2500
119	15,21	5,94	29,71	10,8	2500
120	16,731	6,534	32,681	11,88	2500
121	15,21	5,94	29,71	10,8	2500
122	14,4495	5,643	28,2245	10,26	2500
123	16,731	6,534	32,681	11,88	2500
124	13,689	5,346	26,739	9,72	2500
125	15,9705	6,237	31,1955	11,34	2500
126	15,21	5,94	29,71	10,8	2500
127	15,21	5,94	29,71	10,8	2500
128	14,4495	5,643	28,2245	10,26	2500
129	15,21	5,94	29,71	10,8	2500
130	13,689	5,346	26,739	9,72	2500
131	19,25	5,346	26,84	11,44	1350
132	17,5	4,86	24,4	10,4	1350
133	16,625	4,617	23,18	9,88	1350
134	18,375	5,103	25,62	10,92	1350
135	17,5	4,86	24,4	10,4	1350
136	18,375	5,103	25,62	10,92	1350
137	18,375	5,103	25,62	10,92	1350
138	19,25	5,346	26,84	11,44	1350
139	19,25	5,346	26,84	11,44	1350
140	18,375	5,103	25,62	10,92	1350
141	15,77	4,617	30,59	38,38	2500
142	17,43	5,103	33,81	42,42	2500
143	16,6	4,86	32,2	40,4	2500
144	17,43	5,103	33,81	42,42	2500
145	14,94	4,374	28,98	36,36	2500
146	14,94	4,374	28,98	36,36	2500
147	18,26	5,346	35,42	44,44	2500
148	15,77	4,617	30,59	38,38	2500

149	17,43	5,103	33,81	42,42	2500
150	17,43	5,103	33,81	42,42	2500
151	19,25	5,346	29,15	12,32	1350
152	17,5	4,86	26,5	11,2	1350
153	16,625	4,617	25,175	10,64	1350
154	19,25	5,346	29,15	12,32	1350
155	19,25	5,346	29,15	12,32	1350
156	18,375	5,103	27,825	11,76	1350
157	15,75	4,374	23,85	10,08	1350
158	15,75	4,374	23,85	10,08	1350
159	18,375	5,103	27,825	11,76	1350
160	15,75	4,374	23,85	10,08	1350
161	17,7765	5,103	28,77	11,76	2500
162	15,237	4,374	24,66	10,08	2500
163	15,237	4,374	24,66	10,08	2500
164	16,93	4,86	27,4	11,2	2500
165	17,7765	5,103	28,77	11,76	2500
166	18,623	5,346	30,14	12,32	2500
167	15,237	4,374	24,66	10,08	2500
168	16,93	4,86	27,4	11,2	2500
169	17,7765	5,103	28,77	11,76	2500
170	18,623	5,346	30,14	12,32	2500
171	16,0835	4,617	26,03	10,64	2500
172	16,93	4,86	27,4	11,2	2500
173	18,623	5,346	30,14	12,32	2500
174	15,237	4,374	24,66	10,08	2500
175	16,93	4,86	27,4	11,2	2500
176	15,45	4,86	27,63	11,4	2500
177	16,995	5,346	30,393	12,54	2500
178	15,45	4,86	27,63	11,4	2500
179	14,6775	4,617	26,2485	10,83	2500
180	15,45	4,86	27,63	11,4	2500
181	13,905	4,374	24,867	10,26	2500
182	13,905	4,374	24,867	10,26	2500
183	15,45	4,86	27,63	11,4	2500
184	15,45	4,86	27,63	11,4	2500
185	16,2225	5,103	29,0115	11,97	2500
186	14,6775	4,617	26,2485	10,83	2500
187	14,84	5,4	26,3	11,4	1350
188	14,84	5,4	26,3	11,4	1350
189	14,098	5,13	24,985	10,83	1350

190	16,324	5,94	28,93	12,54	1350
191	14,84	5,4	26,3	11,4	1350
192	15,582	5,67	27,615	11,97	1350
193	15,582	5,67	27,615	11,97	1350
194	13,356	4,86	23,67	10,26	1350
195	15,582	5,67	27,615	11,97	1350
196	16,324	5,94	28,93	12,54	1350
197	14,098	5,13	24,985	10,83	1350
198	14,098	5,13	24,985	10,83	1350
199	13,356	4,86	23,67	10,26	1350
200	20,6115	5,67	25,0005	10,5	2000
201	19,63	5,4	23,81	10	2000
202	18,6485	5,13	22,6195	9,5	2000
203	21,593	5,94	26,191	11	2000
204	19,63	5,4	23,81	10	2000
205	18,6485	5,13	22,6195	9,5	2000
206	17,667	4,86	21,429	9	2000
207	20,6115	5,67	25,0005	10,5	2000
208	17,667	4,86	21,429	9	2000
209	17,667	4,86	21,429	9	2000
210	20,6115	5,67	25,0005	10,5	2000
211	19,327	5,94	26,235	11	2000
212	15,813	4,86	21,465	9	2000
213	16,6915	5,13	22,6575	9,5	2000
214	15,813	4,86	21,465	9	2000
215	17,57	5,4	23,85	10	2000
216	15,813	4,86	21,465	9	2000
217	19,327	5,94	26,235	11	2000
218	16,6915	5,13	22,6575	9,5	2000
219	15,813	4,86	21,465	9	2000
220	15,813	4,86	21,465	9	2000
221	18,4485	5,67	25,0425	10,5	2000
222	16,6915	5,13	22,6575	9,5	2000
223	16,6915	5,13	22,6575	9,5	2000
224	19,327	5,94	26,235	11	2000
225	17,6505	5,67	28,77	11,34	2000
226	16,81	5,4	27,4	10,8	2000
227	16,81	5,4	27,4	10,8	2000
228	18,491	5,94	30,14	11,88	2000
229	16,81	5,4	27,4	10,8	2000
230	18,491	5,94	30,14	11,88	2000

231	16,81	5,4	27,4	10,8	2000
232	15,9695	5,13	26,03	10,26	2000
233	18,491	5,94	30,14	11,88	2000
234	18,491	5,94	30,14	11,88	2000
235	18,48	5,67	20,79	32,34	1350
236	15,84	4,86	17,82	27,72	1350
237	16,72	5,13	18,81	29,26	1350
238	16,72	5,13	18,81	29,26	1350
239	19,36	5,94	21,78	33,88	1350
240	13,662	4,374	25,767	9,72	2000
241	13,662	4,374	25,767	9,72	2000
242	16,698	5,346	31,493	11,88	2000
243	15,939	5,103	30,0615	11,34	2000
244	13,662	4,374	25,767	9,72	2000
245	15,18	4,86	28,63	10,8	2000
246	16,698	5,346	31,493	11,88	2000
247	16,698	5,346	31,493	11,88	2000
248	15,18	4,86	28,63	10,8	2000
249	15,18	4,86	28,63	10,8	2000
250	13,662	4,374	25,767	9,72	2000
251	15,75	5,103	26,04	42,42	2500
252	16,5	5,346	27,28	44,44	2500
253	15	4,86	24,8	40,4	2500
254	15,75	5,103	26,04	42,42	2500
255	15,75	5,103	26,04	42,42	2500
256	13,5	4,374	22,32	36,36	2500
257	14,25	4,617	23,56	38,38	2500
258	14,25	4,617	23,56	38,38	2500
259	15	4,86	24,8	40,4	2500
260	13,5	4,374	22,32	36,36	2500
261	15	4,86	24,8	40,4	2500
262	13,5	4,374	22,32	36,36	2500
263	16,5	5,346	27,28	44,44	2500
264	16,5	5,346	27,28	44,44	2500
265	15,75	5,103	26,04	42,42	2500
266	15,8445	5,103	29,295	10,92	2300
267	14,3355	4,617	26,505	9,88	2300
268	14,3355	4,617	26,505	9,88	2300
269	14,3355	4,617	26,505	9,88	2300
270	15,09	4,86	27,9	10,4	2300
271	16,599	5,346	30,69	11,44	2300

272	13,581	4,374	25,11	9,36	2300
273	15,8445	5,103	29,295	10,92	2300
274	15,8445	5,103	29,295	10,92	2300
275	14,3355	4,617	26,505	9,88	2300
276	15,8445	5,103	29,295	10,92	2300
277	17,6	5,346	29,7	11,44	2300
278	17,6	5,346	29,7	11,44	2300
279	17,6	5,346	29,7	11,44	2300
280	17,6	5,346	29,7	11,44	2300
281	16	4,86	27	10,4	2300
282	17,16	6,534	27,5	11	2000
283	16,38	6,237	26,25	10,5	2000
284	14,82	5,643	23,75	9,5	2000
285	15,6	5,94	25	10	2000
286	15,6	5,94	25	10	2000
287	15,6	5,94	25	10	2000
288	14,82	5,643	23,75	9,5	2000
289	16,38	6,237	26,25	10,5	2000
290	14,82	5,643	23,75	9,5	2000
291	14,82	5,643	23,75	9,5	2000
292	16,4	5,94	20,6	10	2300
293	16,4	5,94	20,6	10	2300
294	18,04	6,534	22,66	11	2300
295	14,76	5,346	18,54	9	2300
296	18,04	6,534	22,66	11	2300
297	16,2225	6,237	28,539	10,5	2000
298	15,45	5,94	27,18	10	2000
299	16,2225	6,237	28,539	10,5	2000
300	14,6775	5,643	25,821	9,5	2000
301	14,6775	5,643	25,821	9,5	2000
302	13,905	5,346	24,462	9	2000
303	14,6775	5,643	25,821	9,5	2000
304	16,995	6,534	29,898	11	2000
305	14,6775	5,643	25,821	9,5	2000
306	15,45	5,94	27,18	10	2000
307	14,6775	5,643	25,821	9,5	2000
308	12,852	5,346	23,4	10,08	2000
309	15,708	6,534	28,6	12,32	2000
310	13,566	5,643	24,7	10,64	2000
311	14,994	6,237	27,3	11,76	2000
312	14,28	5,94	26	11,2	2000

313	15,708	6,534	28,6	12,32	2000
314	14,28	5,94	26	11,2	2000
315	15,708	6,534	28,6	12,32	2000
316	15,708	6,534	28,6	12,32	2000
317	14,994	6,237	27,3	11,76	2000
318	13,566	5,643	24,7	10,64	2000
319	14,28	5,94	26	11,2	2000
320	12,852	5,346	23,4	10,08	2000
321	14,28	5,94	26	11,2	2000
322	14,589	5,346	24,363	32,58	2700
323	17,0205	6,237	28,4235	38,01	2700
324	14,589	5,346	24,363	32,58	2700
325	17,831	6,534	29,777	39,82	2700
326	15,3995	5,643	25,7165	34,39	2700
327	17,831	6,534	29,777	39,82	2700
328	14,589	5,346	24,363	32,58	2700
329	14,589	5,346	24,363	32,58	2700
330	17,831	6,534	29,777	39,82	2700
331	14,589	5,346	24,363	32,58	2700
332	17,0205	6,237	28,4235	38,01	2700
333	17,0205	6,237	28,4235	38,01	2700
334	16,21	5,94	27,07	36,2	2700
335	17,0205	6,237	28,4235	38,01	2700
336	15,435	6,237	24,675	11,76	2300
337	15,435	6,237	24,675	11,76	2300
338	13,23	5,346	21,15	10,08	2300
339	13,23	5,346	21,15	10,08	2300
340	14,7	5,94	23,5	11,2	2300
341	13,965	5,643	22,325	10,64	2300
342	13,965	5,643	22,325	10,64	2300
343	14,7	5,94	23,5	11,2	2300
344	13,23	5,346	21,15	10,08	2300
345	14,7	5,94	23,5	11,2	2300
346	17,43	5,103	22,26	11,34	2500
347	18,26	5,346	23,32	11,88	2500
348	18,26	5,346	23,32	11,88	2500
349	14,94	4,374	19,08	9,72	2500
350	14,94	4,374	19,08	9,72	2500
351	16,6	4,86	21,2	10,8	2500
352	14,94	4,374	19,08	9,72	2500
353	15,77	4,617	20,14	10,26	2500

354	18,26	5,346	23,32	11,88	2500
355	18,26	5,346	23,32	11,88	2500
356	16,6	4,86	21,2	10,8	2500
357	18,26	5,346	23,32	11,88	2500
358	15,77	4,617	20,14	10,26	2500
359	15,77	4,617	20,14	10,26	2500
360	16,6	4,86	21,2	10,8	2500
361	19,03	5,346	31,57	11,88	2500
362	16,435	4,617	27,265	10,26	2500
363	19,03	5,346	31,57	11,88	2500
364	17,3	4,86	28,7	10,8	2500
365	16,435	4,617	27,265	10,26	2500
366	17,3	4,86	28,7	10,8	2500
367	17,3	4,86	28,7	10,8	2500
368	17,3	4,86	28,7	10,8	2500
369	18,165	5,103	30,135	11,34	2500
370	17,3	4,86	28,7	10,8	2500
371	11,875	4,617	26,125	38,76	2300
372	13,75	5,346	30,25	44,88	2300
373	13,125	5,103	28,875	42,84	2300
374	11,875	4,617	26,125	38,76	2300
375	13,75	5,346	30,25	44,88	2300
376	12,5	4,86	27,5	40,8	2300
377	16,896	5,346	29,392	37,62	2700
378	13,824	4,374	24,048	30,78	2700
379	14,592	4,617	25,384	32,49	2700
380	14,592	4,617	25,384	32,49	2700
381	13,824	4,374	24,048	30,78	2700
382	16,128	5,103	28,056	35,91	2700
383	16,896	5,346	29,392	37,62	2700
384	16,896	5,346	29,392	37,62	2700
385	14,592	4,617	25,384	32,49	2700
386	13,824	4,374	24,048	30,78	2700
387	13,824	4,374	24,048	30,78	2700
388	13,5	4,374	23,967	10,08	2700
389	15,75	5,103	27,9615	11,76	2700
390	15	4,86	26,63	11,2	2700
391	16,5	5,346	29,293	12,32	2700
392	16,5	5,346	29,293	12,32	2700
393	15,75	5,103	27,9615	11,76	2700
394	15,75	5,103	27,9615	11,76	2700

395	16,5	5,346	29,293	12,32	2700
396	15	4,86	26,63	11,2	2700
397	16,5	5,346	29,293	12,32	2700
398	15,75	5,103	27,9615	11,76	2700
399	15,295	4,617	24,035	10,64	2300
400	17,71	5,346	27,83	12,32	2300
401	14,49	4,374	22,77	10,08	2300
402	17,71	5,346	27,83	12,32	2300
403	15,295	4,617	24,035	10,64	2300
404	14,49	4,374	22,77	10,08	2300
405	17,71	5,346	27,83	12,32	2300
406	17,71	5,346	27,83	12,32	2300
407	16,905	5,103	26,565	11,76	2300
408	16,1	4,86	25,3	11,2	2300
409	17,16	5,94	23,683	12,32	2700
410	14,04	4,86	19,377	10,08	2700
411	14,04	4,86	19,377	10,08	2700
412	14,04	4,86	19,377	10,08	2700
413	17,16	5,94	23,683	12,32	2700
414	14,82	5,13	20,4535	10,64	2700
415	16,38	5,67	22,6065	11,76	2700
416	16,38	5,67	22,6065	11,76	2700
417	16,38	5,67	22,6065	11,76	2700
418	14,04	4,86	19,377	10,08	2700
419	14,82	5,13	20,4535	10,64	2700
420	16,38	5,67	22,6065	11,76	2700
421	16,38	5,67	22,6065	11,76	2700
422	16,38	5,67	22,6065	11,76	2700
423	15,6	5,4	21,53	11,2	2700
424	15,8	5,4	23	37,8	2300
425	16,59	5,67	24,15	39,69	2300
426	16,59	5,67	24,15	39,69	2300
427	17,38	5,94	25,3	41,58	2300
428	15,8	5,4	23	37,8	2300
429	16,39	5,94	28,6	11,88	2500
430	13,41	4,86	23,4	9,72	2500
431	16,39	5,94	28,6	11,88	2500
432	15,645	5,67	27,3	11,34	2500
433	14,9	5,4	26	10,8	2500
434	16,39	5,94	28,6	11,88	2500
435	16,39	5,94	28,6	11,88	2500

436	16,39	5,94	28,6	11,88	2500
437	16,39	5,94	28,6	11,88	2500
438	14,9	5,4	26	10,8	2500
439	15,39	5,13	23,56	9,88	2300
440	15,39	5,13	23,56	9,88	2300
441	17,82	5,94	27,28	11,44	2300
442	15,39	5,13	23,56	9,88	2300
443	17,82	5,94	27,28	11,44	2300
444	16,2	5,4	24,8	10,4	2300
445	14,58	4,86	22,32	9,36	2300
446	14,58	4,86	22,32	9,36	2300
447	16,2	5,4	24,8	10,4	2300
448	17,82	5,94	27,28	11,44	2300
449	14,421	5,13	27,113	9,88	2700
450	14,421	5,13	27,113	9,88	2700
451	16,698	5,94	31,394	11,44	2700
452	15,939	5,67	29,967	10,92	2700
453	13,662	4,86	25,686	9,36	2700
454	16,698	5,94	31,394	11,44	2700
455	14,421	5,13	27,113	9,88	2700
456	14,421	5,13	27,113	9,88	2700
457	16,698	5,94	31,394	11,44	2700
458	15,18	5,4	28,54	10,4	2700
459	13,662	4,86	25,686	9,36	2700
460	14,763	5,67	31,395	11,34	2700
461	12,654	4,86	26,91	9,72	2700
462	15,466	5,94	32,89	11,88	2700
463	15,466	5,94	32,89	11,88	2700
464	14,763	5,67	31,395	11,34	2700
465	13,357	5,13	28,405	10,26	2700
466	12,654	4,86	26,91	9,72	2700
467	15,466	5,94	32,89	11,88	2700
468	14,06	5,4	29,9	10,8	2700
469	12,654	4,86	26,91	9,72	2700
470	15,466	5,94	32,89	11,88	2700
471	15,466	5,94	32,89	11,88	2700
472	14,06	5,4	29,9	10,8	2700
473	15,466	5,94	32,89	11,88	2700
474	14,763	5,67	31,395	11,34	2700
475	14,763	5,67	31,395	11,34	2700
476	17,42	4,86	30,35	10,4	2700

477	19,162	5,346	33,385	11,44	2700
478	16,549	4,617	28,8325	9,88	2700
479	18,291	5,103	31,8675	10,92	2700
480	16,549	4,617	28,8325	9,88	2700
481	17,42	4,86	30,35	10,4	2700
482	16,549	4,617	28,8325	9,88	2700
483	19,162	5,346	33,385	11,44	2700
484	15,678	4,374	27,315	9,36	2700
485	17,42	4,86	30,35	10,4	2700
486	15,678	4,374	27,315	9,36	2700
487	19,162	5,346	33,385	11,44	2700
488	18,291	5,103	31,8675	10,92	2700
489	18,291	5,103	31,8675	10,92	2700
490	17,5	4,86	28,8	10,4	2700
491	19,25	5,346	31,68	11,44	2700
492	17,5	4,86	28,8	10,4	2700
493	16,625	4,617	27,36	9,88	2700
494	15,75	4,374	25,92	9,36	2700
495	18,375	5,103	30,24	10,92	2700
496	17,5	4,86	28,8	10,4	2700
497	19,25	5,346	31,68	11,44	2700
498	17,5	4,86	28,8	10,4	2700
499	15,75	4,374	25,92	9,36	2700
500	15,75	4,374	25,92	9,36	2700